1994

Erosion effects on TVC vane heat transfer characteristics

Gardner, Steven R.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

EROSION EFFECTS ON
TVC VANE HEAT
TRANSFER CHARACTERISTICS

by

Steven R. Gardner

March, 1994

Thesis Advisor: Morris Driels

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE March 1994	3. REPORT TYPE AND DATES COVERED Master's Thesis
4. TITLE AND SUBTITLE Erosion Effects on TVC Vane Heat Transfer Characteristics		5. FUNDING NUMBERS
6. AUTHOR(S) Steven R. Gardner		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.		
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.		12b. DISTRIBUTION CODE *A

13.

ABSTRACT *(maximum 200 words)*

This work describes the effects of erosion on the heat transfer characteristics on thrust vector control vanes exposed to aluminized propellant exhaust flows. This was accomplished using an inverse heat transfer parameter identification of quarter scale models. The model is based on a four node lumped parameter system with two heat energy inputs. The erosion is modeled as decreasing the geometric dimensions linearly as a function of time and percentage of aluminum in the propellant. Excellent agreement was found between experimental and model temperature profiles. The heat transfer coefficients of the vanes were found to decrease with increasing erosion rates.

14. SUBJECT TERMS Thrust vector control, erosion, ablation, parametric system identification, erosion front modeling, aluminized propellant			15. NUMBER OF PAGES 116
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

Approved for public release; distribution is unlimited.

Erosion Effects on TVC Vane Heat Transfer Characteristics

by

Steven R. Gardner
Lieutenant, United States Navy
B.S., Worcester Polytechnic Institute, 1988

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 1994

Author: [redacted]

Steven R. Gardner

Approved by: [redacted]

Morris Driels, Thesis Advisor

[redacted]

Matthew Kelleher, Chairman
Department of Mechanical Engineering

ABSTRACT

This work describes the effects of erosion on the heat transfer characteristics on thrust vector control vanes exposed to aluminized propellant exhaust flows. This was accomplished using an inverse heat transfer parameter identification of quarter scale models. The model is based on a four node lumped parameter system with two heat energy inputs. The erosion is modeled as decreasing the geometric dimensions linearly as a function of time and the percentage of aluminum in the propellant. Excellent agreement was found between experimental and model temperature profiles. The heat transfer coefficients of the vanes were found to decrease with increasing erosion rates.

C/113
0.2

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

This thesis is a continuation of work done by the Naval Air Warfare Center Weapons Division (NAWCWPNS) and thesis work at the Naval Postgraduate School (NPS) to provide a better understanding of the heat transfer characteristics of jet vanes used for thrust vector control (TVC) of vertical launch missiles. This is accomplished using an inverse heat transfer parameter identification of quarter scale replicas which can be used to find full scale results.

Thrust vector control is a process by which jet vanes are inserted into the exhaust plume of a missile to control the flight path prior to the missile obtaining the required velocity for the external control surfaces to take effect. [Ref. 1] A schematic for the TVC system is shown in Figure 1.

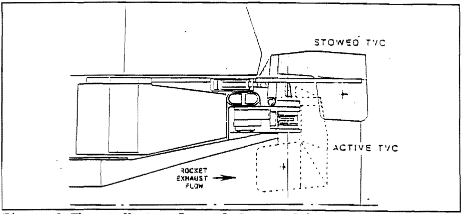


Figure 1 Thrust Vector Control System Schematic

Due to the harsh thermal environment that the vanes are exposed to, a better understanding of the heat transfer processes which take place will help in the improved design of jet vanes. This will lead to longer operation and the ability to use propellants that burn hotter and use a higher percentage of aluminum for greater momentum flux and better performance. [Ref. 2:p. 1]

There are five basic steps in determining the heat transfer characteristics of the vane:

1. Develop a mathematical model of the heat transfer processes which take place in the vane. It is expressed in terms of a number of physical constants, some of which are known, some of which are to be determined. [Ref. 3:p. 1]

2. Gather experimental data in the form of temperature-time data at selected locations on the vane.

3. Compare the predicted and experimental temperature-time data.

4. Use the differences between the simulated and actual temperatures to drive a systematic adjustment of unknown model parameters in an optimization routine. The process is repeated until the experimental and theoretical data differences are minimized in a least-squares sense. [Ref. 3:p. 2]

5. Calculate the heat transfer parameters of the system using the physical parameters of the model which give the best estimate of the actual behavior.

Previous work has concentrated on using parametric system identification to validate the use of full and quarter scale models to predict the heat transfer characteristics for full scale vanes in a non-erosive environment. The research in this report extends the quarter scale model to an erosive environment.

II. THEORY

A. BACKGOUND

1. Physical Description

The main pieces of equipment used in the experimental tests are the rocket motor and the jet vanes. The rocket motor is set up to provide a constant thrust-time profile. The propellants used in the motor are aluminized hydroxyl-terminated polybutadiene (HTPB) with either 0%, 9%, or 18% Al by weight. The jet vanes are made from pressed and sintered tungstan powder that is infiltrated by 10% copper by weight. There are four vanes for each motor. The experimental setup is shown in Figure 2. [Ref. 2:p. 1,2]

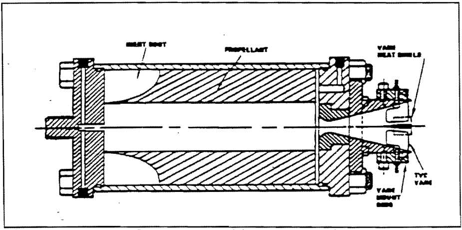


Figure 2 Vane and Motor Assembly for Experimental Tests

The experimental tests are conducted as either full or quarter scale. The quarter scale tests have several advantages. Most important is the cost savings over a full scale test. The reduced size of the motor, vane and test equipment account for much of the savings. [Ref. 4:p. 15,16] The biggest disadvantage of the quarter scale vane comes in the placement of the thermocouples. Whereas in the full scale vane the thermocouples can be placed inside the vane, for the quarter scale vane the thermocouples must be placed on the vane shaft. The thermocouple placement is contrasted in Figure 3.

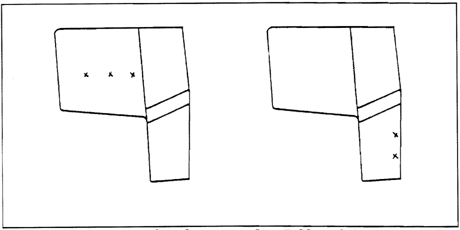


Figure 3 Thermocouple Placement for Full and Quarter Scale

2. Basic Modeling

In order to predict the thermal response of the jet vane, a simple model had to be developed. The model had to

consider the physical characteristics of the vane and the heat transfer processes that were taking place.

The physical quantities can be broken into two categories: material and geometric. The material properties considered were the vane density, thermal conductivity, and specific heat. The geometric properties considered were the conductive lengths, cross sectional and surface areas and volume.

The heat transfer processes considered were convection at the surface of the vane and conduction of heat through the vane.

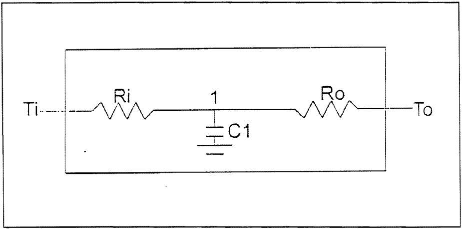


Figure 4 Thermal Energy Node Model

Heat transfer in the vane is modeled by applying the law of conservation of energy. Energy balance equations can be derived using a model consisting of thermal resistances and

capacitances driven by the temperature difference between the nodes. The energy balance for Figure 4 is,

$$\dot{T}_1=\frac{T_i}{C_1R_i}-\frac{T_1}{C_1R_i}-\frac{T_1}{C_1R_o}+\frac{T_o}{C_1R_o} \tag{1}$$

where $T_i>T_1>T_o$. The convective resistance is found by,

$$R=\frac{1}{hA_s} \tag{2}$$

where h is the convective heat transfer coefficient and A_s is the surface area. The conductive resistance is found by,

$$R=\frac{L}{kA_x} \tag{3}$$

where L is the conductive length, k is the thermal conductivity and A_x is the cross sectional area. The thermal capacitance is given by,

$$C=\rho VC_p \tag{4}$$

where ρ is the material density, V is the volume and C_p is the material specific heat.

3. Lumped Parameter

The nodes of the basic model lend itself to dividing the vane into different sections, or lumps. For the full

scale model, the vane was geometrically divided into three separate sections: the tip, fin and shaft. A node is located at the center of each section. The sections are defined as shown in Figure 5. For the quarter scale model, a fourth node was added at the mount to account for the different thermocouple placement.

4. PSI Process

A simple model was needed that could easily be changed for different materials, geometries and exhaust conditions. This lead to parametric system identification (PSI). PSI is a computer based procedure where the parameters of a model are changed until a best fit approximation in a least squares sense to experimental data is obtained. [Ref. 5:p. 6]

Parameter identification has several advantages over the other modeling choice, computational fluid dynamics (CFD). Creating a mathmatical model of the vane using CFD is almost impossible due to the complexity of the exhaust flow. The jet vane must operate in a high temperature, three-dimensional, turbulent, compressible supersonic flow. [Ref. 5:p. 3,4] PSI ignores these complexities and focuses on the end result. This makes PSI not only simpler, but the information that comes out of the PSI model can easily be used in improving the design. PSI also handles nonlinear conditions such as ablation. [Ref. 6: p.2]

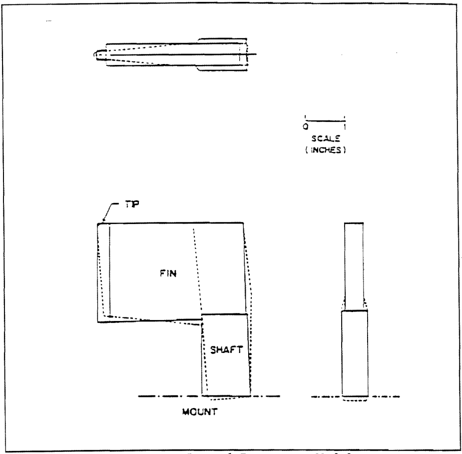


Figure 5 Jet Vane as a Lumped Parameter Model (Actual Design Indicated by Dotted Lines)

The simple three node model shown in Figure 6 can serve as baseline for other models. The model is driven by two heat sources, represented by temperatures T_{R1} and T_{R2}, which are the stagnation and free stream recovery temperatures, respectively. The temperature T_{R1} heats the vane through convective heat transfer at node one through the thermal

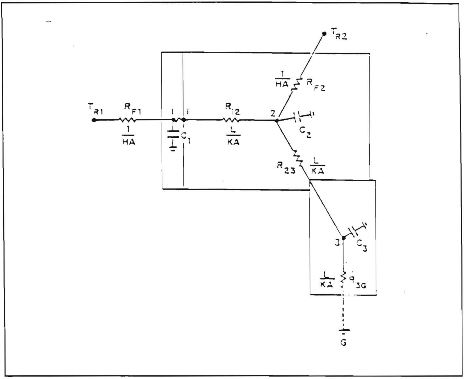


Figure 6 Three Node Jet Vane Model

resistance R_{F1}, and stores the energy as a thermal capacitance in C_1. The same process occurs at node two with recovery temperature T_{R2}, thermal resistance R_{F2}, and thermal capacitance C_2. Node three stores energy in thermal capacitance C_3 and is connected to ground through thermal resistance R_{3G}. All nodes are coupled by conductive resistances. [Ref. 6:p. 4] Applying the law of conservation of energy to the system leads to the following equations:

$$\dot{T}_1 = -\frac{T_1}{C_1 R_{F1}} - \frac{T_1}{C_1 R_{12}} + \frac{T_2}{C_1 R_{12}} + \frac{T_{R1}}{C_1 R_{F1}} \tag{5}$$

$$\dot{T}_2 = \frac{T_1}{C_2 R_{12}} - \frac{T_2}{C_2 R_{F2}} - \frac{T_2}{C_2 R_{12}} - \frac{T_2}{C_2 R_{23}} + \frac{T_3}{C_2 R_{23}} + \frac{T_{R2}}{C_2 R_{F2}} \tag{6}$$

$$\dot{T}_3 = \frac{T_2}{C_3 R_{23}} - \frac{T_3}{C_3 R_{23}} - \frac{T_3}{C_3 R_{3G}} \tag{7}$$

letting,

$$a_{12} = \frac{1}{C_1 R_{12}} \qquad a_{21} = \frac{1}{C_2 R_{12}} \tag{8}$$

$$a_{23} = \frac{1}{C_2 R_{23}} \qquad a_{32} = \frac{1}{C_3 R_{23}} \tag{9}$$

$$a_{3G} = \frac{1}{C_3 R_{3G}} \qquad b_{11} = \frac{1}{C_1 R_{F1}} \tag{10}$$

$$b_{22} = \frac{1}{C_2 R_{F2}} \tag{11}$$

Combining coefficients at the same temperatures gives,

$$a_{11} = a_{12} + b_{11} \tag{12}$$

$$a_{22} = a_{21} + a_{23} + b_{22} \tag{13}$$

$$a_{33} = a_{32} + a_{3G} \tag{14}$$

Rewriting the equations,

$$\dot{T}_1 = -a_{11} T_1 + a_{12} T_2 + b_{11} T_{R1} \tag{15}$$

$$\dot{T}_2 = a_{21}T_2 - a_{22}T_2 + a_{23}T_3 + b_{22}T_{R2} \quad \textbf{(16)}$$

$$\dot{T}_3 = a_{32}T_2 - a_{33}T_3 \quad \textbf{(17)}$$

Rewriting into state-space form, T = AT + Bu, or

$$\begin{matrix} \dot{T}_1 \\ \dot{T}_2 \\ \dot{T}_3 \end{matrix} = \begin{matrix} -a_{11} & a_{21} & 0 \\ a_{21} & -a_{22} & a_{23} \\ 0 & a_{32} & -a_{33} \end{matrix} \begin{matrix} T_1 \\ T_2 \\ T_3 \end{matrix} + \begin{matrix} b_{11} & 0 & 0 \\ 0 & b_{22} & 0 \\ 0 & 0 & 0 \end{matrix} \begin{matrix} T_{R1} \\ T_{R2} \\ 0 \end{matrix} \quad \textbf{(18)}$$

The energy balance equations are a set of linear, ordinary differential equations which can be readily solved on a computer. This was done in a Fortran program using an IMSL subroutine called DIVPRK. DIVPRK solves a double precision initial value problem for ordinary differential equations using fifth-order and sixth-order Runge-Kutta-Verner methods. DIVPRK requires a user supplied subroutine called FCN which defines the set of equations to be solved.

The main program containing DIVPRK and FCN is called SIM.FOR, and simulates the temperatures of the three node model. The model is driven by an input vector u which is the product of the recovery temperatures T_{R1} and T_{R2} and a step function simulating the thrust. Physical and geometric data was used to calculate the internal thermal conductive resistances and capacitances which lead to coefficients in the A matrix. Since the inputs at nodes one and two from

convection and node three from ground are unknown, values for these resulting coefficients must be guessed. The output of the program is temperature-time data which is written to a data file called TEMP.MAT. This data can then be read into MATLAB and plotted. The purpose is to try to match calculated temperatures with known experimental temperature data at node two and validate the numerical approach. The results are shown in Figure 7. Although the node two temperatures are close, they are not identical. By extending the program to include an optimizer that could adjust the unknown A and B coefficients, a closer approximation could be found.

This was done in a Fortran program called NODE3.FOR. It is in this parameter identification, or PID, program that the differential equations are set up and solved. First, physical and geometric data is read in from a data file called INPUT.DAT. This information is used to calculate the internal thermal conductive resistances and capactitances which lead to coefficients in the A matrix. Since the inputs at nodes one and two from convection and node three from ground are unknown, the resulting unknown coefficients from the A and B matrices are sent to the optimizer as variables to be found. The optimizer used is an IMSL routine called DBCLSF which uses a modified Levenberg-Marquardt method and an active set strategy to minimize an error in a least-squares sense subject to simple constraints placed on the variables by the user. DBCLSF calls a user written subroutine called TEMP that

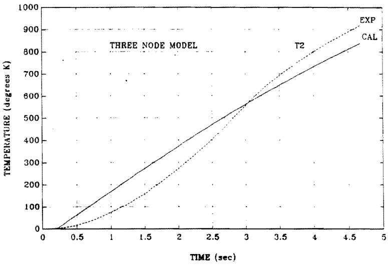


Figure 7 Simulated and Experimental Node Temperatures

calculates the temperature-time history using the current parameters supplied by DBCLSF called from the PID program. It does this through DIVPRK and FCN. Once the temperature-time history is calculated, an error function is returned to DBCLSF based on the differences between predicted and experimental temperature-time histories. The optimizer then adjusts the

unknown parameters and the process repeats until certain convergence criteria is met.

B. PREVIOUS MODELS

Work on the jet vane thermal model began at the Naval Postgraduate School (NPS) by Nunn and Kelleher [Ref. 7] in 1986. Further development of the model was continued by Nunn [Ref. 5] and Hatzenbuehler. [Ref. 8] Hatzenbuehler was able to create a four node quarter scale model using PSI procedures and a computer software package called Matrix X. Reno [Ref. 1] followed Hatzenbuehler and refined the four node model and attempted to compare the quarter scale results to full scale vanes, but was unsuccessful. More recent work has been done by Parker [Ref. 4]. He obtained good results using a full scale model of the jet vane. He also looked more closely at the scaling of the models and the applicability of quarter scale results to full scale vanes. He also found that existing quarter scale models did not provide an accurate picture of the heat transfer processes in the full scale vanes.

C. THREE NODE FULL SCALE MODEL

Parker's five node full scale model was reduced to a three node full scale model to investigate whether the three fin nodes could be reduced to one node and obtain the same

results. Parker's five node full scale model is shown in Figure 8.

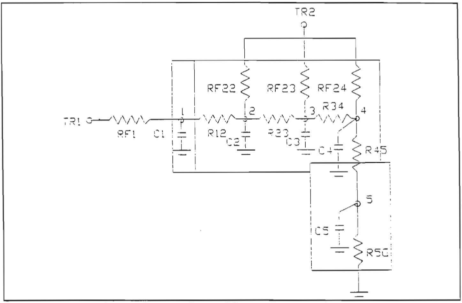


Figure 8 Parker Five Node Full Scale Model

The three node model was driven using the geometric data given in Table 1 and the following material data: ρ = 18310 kg/m^3, k = 173 W/mK, and C_p = 146 J/kgK. The recovery temperatures used to drive the system were T_{R1} = 2670 K and T_{R2} = 2570 K. [Ref. 6:p. 7,8] These temperatures were contained in the input vector u, whose values were the product of the thrust recovery temperature and a step function simulating the thrust function.

Table I GEOMETRIC DATA FOR FULL SCALE VANES

	tip	to	vane	to	shaft
V, cm^4	2.6		52.0		23.0
A_y, cm^3		5.9		5.2	
A_l, cm^3	4.35		112.16		
L, cm		5.0		6.0	

The program found the values for b_{11} = 1.0029 and b_{22} = 0.0809. This corresponds to the convection heat transfer coefficients of 16,025 W/m^2K and 1003 W/m^2K at the tip and fin respectively. The ground resistance was found to be 0.0001. These values were found to be reasonably close to those from Parker's five node model. He found b_{11} = 1.3787, b_{23} = 0.0862, and the ground resistance to be 0.0001, while the convection heat transfer coefficients were 22027.5 W/m^2K and 1057 W/m^2K at the tip and fin respectively. [Ref. 4:p. 63] The temperature-time histories for both models are shown in Figure 9.

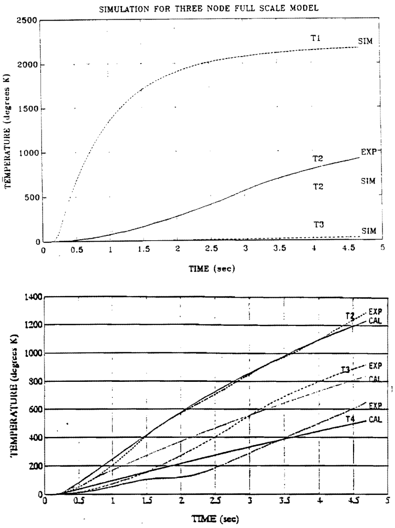


Figure 9 Temperature-time Histories for Three and Five Node Full Scale Models

III. ABLATION EFFECTS

A. ABLATION MODELING

There was erosion in the quarter scale vanes exposed to aluminized propellant exhaust flows. For the 0% aluminized case, only 1% of the vane's mass was lost. But for the 9% and 18% aluminized cases, the loss became much more substantial. For the 9% case, 8% of the vane's mass was lost. For the 18% aluminized case, 50% of the vane's mass was lost. Vane mass loss was found to be nonlinear with the percentage of Al in the propellant. The relationship using an exponential function by an empirical fit was found to be

$$\%mass\ loss = 1.042e^{(0.2173)(\%Al)} \qquad \mathbf{(19)}$$

Vane erosion profiles for the three cases are shown in Figure 10. [Ref. 2:p. 6,7] At least part of this erosion was likely caused by ablation. Ablation is due to the melting of the surface of the vane [Ref. 9:p. 122].

A short FORTRAN program, COEF.FOR, was written to see how the known A matrix coefficients were affected by the mass loss. The geometric dimensions of length, area, and volume were modeled as decreasing linearly as a function of time and percent mass loss. The results for the 9% Al and 18% Al cases are shown in Figure 11.

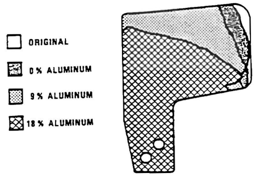


Figure 10 Vane Erosion Profiles

Several trends in Figure 11 are worth noting. The dominent coefficient in both cases is a_{12}. This is expected since

$$a_{12}=\frac{1}{C_1 R_{12}} \tag{20}$$

and C_1 is small due to the small volume at the tip of the vane. Also note that the coefficients are nonlinear over time and the nonlinearity increases with increased mass loss.

B. FOUR NODE QUARTER SCALE MODEL

The erosion present in the quarter scale vanes when the aluminized propellant was used needed to be investigated. A four node quarter scale model had already been derived by

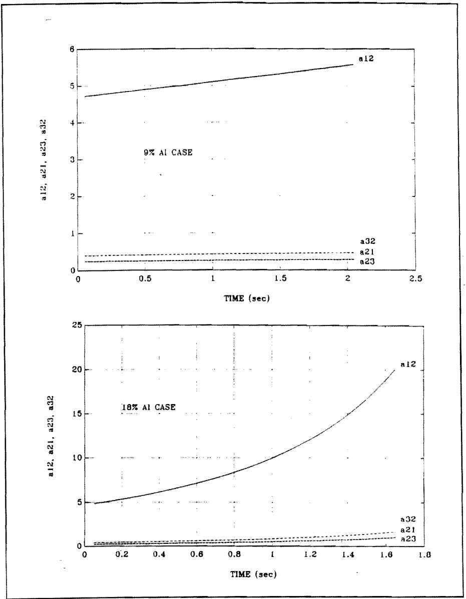


Figure 11 Effect of Erosion on A Matrix Coefficients in the 9% Al and 18% Al Cases

Reno. [Ref. 1] Application of the law of conservation of energy led to the following equations:

$$\dot{T}_1=-a_{11}T_1+a_{12}T_2+b_{11}T_{R1} \tag{21}$$

$$\dot{T}_2=a_{21}T_1-a_{22}T_2+a_{23}T_3+b_{22}T_{R2} \tag{22}$$

$$\dot{T}_3=a_{32}T_2-a_{33}T_3+a_{34}T_4 \tag{23}$$

$$\dot{T}_4=a_{43}T_3-a_{44}T_4 \tag{24}$$

These equations needed to be modified though, since they did not include the effects of erosion. Erosion of the vane caused the geometric dimensions of the vane to change, while the material properties of density, thermal conductivity and specific heat remained constant. The program COEF.FOR modeled the changing geometric dimensions with time. All that was needed was to attach COEF.FOR to the main PID program as a subprogram.

The other aspect of interest in the cases with aluminized propellant was whether the convective heat transfer coefficients were time varient. Once the values of b_{11} and b_{22} are found in the PID program, the program COEF.FOR can be modified so that the heat transfer coefficients can be calculated at every time step since

$$h_t=\frac{1}{R_{F1}A_{st}} \qquad h_f=\frac{1}{R_{F2}A_{sf}} \tag{25}$$

$$R_{F1}=\frac{1}{b_{11}C_1} \qquad R_{F2}=\frac{1}{b_{22}C_2} \tag{26}$$

and C_1, C_2, A_{st}, and A_{sf} are all time dependant.

C. CONVERGING QUARTER SCALE MODEL WITH ABLATION

1. Case 1: 0% Al in Propellant

For case 1, data was taken for three seconds before thrust began to tailoff. This allowed for 61 temperature-time data points to be taken, or 20 per second. The data points on the vane corresponded to nodes three and four of the model. This data was read into the PID program NODE40.FOR along with the geometric data and the recovery temperatures. In the subroutine FCN, a delay of 0.3 seconds was used to account for the time before the thrust reached its steady state value. The results obtained were excellent; the square root of the sum of the squares of the difference between experimental and model temperatures at nodes three and four was only 1.19 degrees Kelvin. A plot of the experimental and model temperatures is shown in Figure 12.

The values obtained for the unknown variables were a_{34}=0.5376, a_{43}=0.1528, a_{4G}=-0.1651, b_{11}=7.6511, and b_{22}=0.0722. These variables led to resistance values of R_{F1}=1.2221, R_{F2}=6.4795, and R_{4G}=-1.8206. The negative value obtained for R_{4G} indicates heating of the vane from ground. The convection heat transfer coefficients were calculated to be 30405.43

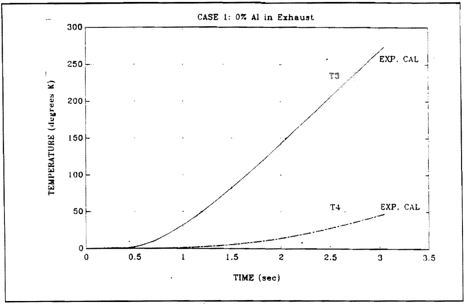


Figure 12 Case 1: Experimental and Model Temperatures Vs. Time

W/m²K and 222.43 W/m²K at the tip and fin respectively.

2. Case 2: 9% Al in Propellant

The same procedure was done for case 2. Temperature-time data was only taken for two seconds before thrust tailoff. A delay of 0.7 seconds was used to account for the time before the thrust reached its steady state value. Again, the results were excellent: the sum of the squares difference was only 0.73 degrees Kelvin. A plot of the experimental and model temperatures is shown in Figure 13.

The values obtained for the unknown variables were a_{34}=-0.2000, a_{43}=3.5088, a_{40}=100.00, b_{11}=3.4427, and b_{22}=0.0837.

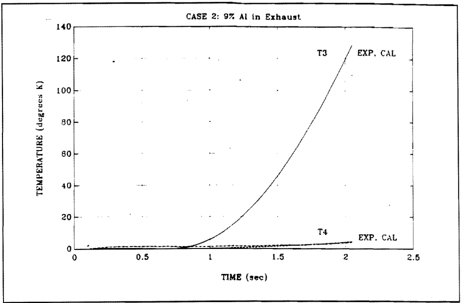


Figure 13 Case 2: Experimental and Model Temperatures Vs. Time

The variables lead to resistance values of R_{F1}=2.9135, R_{F2}=5.9930, and R_{4G}=-0.1989. Again, the negative resistance of R_{4G} indicates heating of the vane from ground. The convection heat transfer coefficients were calculated to be 13354.40 and 251.80 at the tip and fin respectively.

The values for b_{11} and b_{22} found from NODE49.FOR were added to the geometric and material data in COEF.FOR in order to calculate the convective heat transfer coefficients at every time step. The heat transfer coefficient at the tip decreased from an initial value of 13742.78 to the final value of 13354.40. The heat transfer coefficient for the fin decreased from an initial value of 259.17 to the final value

of 251.80. In both cases, there was only a three percent decrease. The coefficients are plotted versus time in Figure 14.

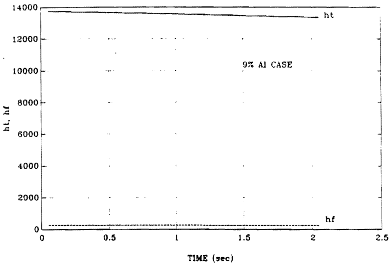


Figure 14 Convective Heat Transfer Coefficients Plotted Vs. Time For Case 2.

3. Case 3: 18% Al in Propellant

The same procedure was done for case 3. Temperature-time data was only taken for 1.6 seconds before the severity of the erosion caused direct plume impingment to the vane shaft. [Ref. 2,p.9] A delay of 0.1 seconds was used to account for the time before the thrust reached its steady state value. Again the results were excellent: the sum of

the squares difference was only 1.52 degrees Kelvin. A plot of the experimental and model temperatures is shown in Figure 15.

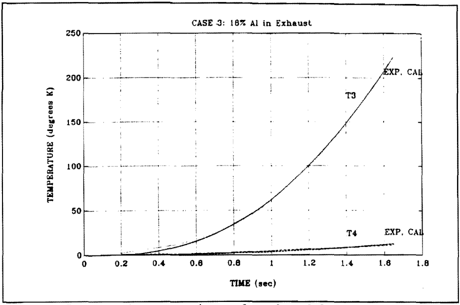


Figure 15 Case 3: Experimental and Model Temperatures Vs. Time

The values obtained for the unknown variables were a_{34}=-0.2000, a_{43}=5.9382, a_{40}=100.00, b_{11}=1.6236, and b_{22}=0.0500. The variables lead to resistance values of R_{F1}=11.7085, R_{F2}=19.0253, and R_{40}=-0.6380. Again, the negative resistance of R_{40} indicating heating from ground. The values of the convection heat transfer coefficients were calculated to be 4786.95 W/m²K and 114.26 W/m²K at the tip and fin respectively.

The values for b_{11} and b_{22} found from NODE418.FOR were added to the geometric and material data in COEF.FOR to again

find the convective heat transfer coefficients at every time step. At the tip, the heat transfer coefficient decreased from an initial value of 6451.38 to the final value of 4786.95. The heat transfer coefficient for the fin also decreased, from an initial value of 154.11 to the final value of 114.26. There was a 26% decrease at both the tip and fin, with the tip showing nonlinearities. The coefficients are plotted in Figure 16.

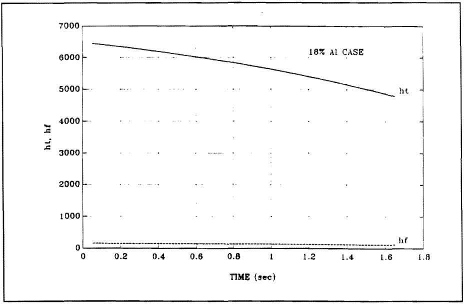


Figure 16 Convective Heat Transfer Coefficients Plotted Vs. Time For Case 3.

D. Erosion Front Modeling

An energy balance equation can be written between the leading edge erosion heat flux, q/A_o, and the heat required to maintain the vane leading edge ablation rate, or

$$\frac{q}{A_o}=S_T\rho_\infty U_\infty C_{P_\infty}(T_{AW}-T_W)=\dot{S}\rho_{LE}F[1+\frac{C(T_M-T_W)}{F}] \qquad \textbf{(27)}$$

where S_T is the Stanton number, T_{AW} is the leading edge recovery temperature, T_W is the vane leading edge temperature, T_M is the melting temperature of the vane material, F is the heat of fusion for tungsten, and C is the heat capacity of tungsten. Also note that

$$S_T\rho_\infty U_\infty C_{P_\infty}=H_{LE} \qquad \textbf{(28)}$$

where H_{LE}, the leading edge convection heat transfer coefficient, is found by a parameter identification program like one of those previously described. [Ref. 10:p. 2,3]

A theoretical erosion rate can be found by manipulating equations (27) and (28)

$$\dot{S}=\frac{H_{LE}(T_{AW}-T_W)}{\rho_{LE}F[1+\frac{C(T_M-T_W)}{F}]} \qquad \textbf{(29)}$$

T_W can be estimated by running a four node simulation model and using the node one temperatures at each time step.

Equation (27) is based upon ablation of the vane, which requires that $T_W > T_M$. Therefore the erosion rate was set equal to zero until T_W reaches T_M. The melting temperature for the vane, which is a 90% tungsten-10% copper alloy by weight, is 3513K. This temperature is higher than T_W for both the 9% and 18% cases, and therefore theoretically the vane should not erode. Since the vane does erode, T_M for the vane was taken as the melting temperature of copper, 1358K. This seemed reasonable since the melting point of copper is lower than that of tungsten.

Once the erosion rate is found, it can then be integrated over the time of the firing to find a theoretical length of the vane eroded. This was done in both the 9% and 18% cases and is shown in Figures 17 and 18. The length of the vane eroded using this method is estimated as 1.1 cm for the 9% case and 2.3 cm for the 18% case. Although the 1.1 cm found for the 9% case is high compared to the 0.4 cm found experimentally, the 2.3 cm found for the 18% case is very close to the 2.5 cm found experimentally.

Equation (29) can also be used to try and validate the use of the melting temperature of copper for T_M. This was done by plotting the vane temperatures found in the simulation programs as a function of the length between nodes one and two, then using the known total length eroded from the experiment to find the apparent melting temperature. The plots for the 9% and 18% cases are shown in Figures 19 and 20.

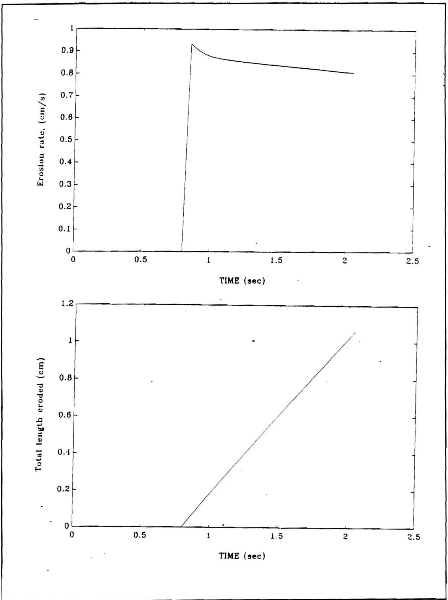


Figure 17 Erosion Rate and Total Length Eroded for the 9% Case

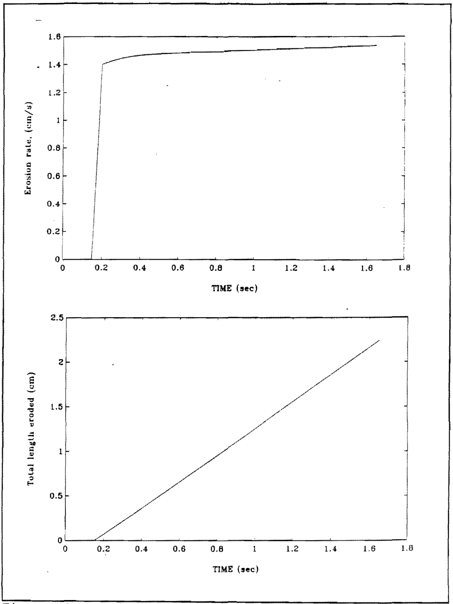


Figure 18 Erosion Rate and Total Length Eroded for the 18% Case

The melting temperature found for the 9% case was 1732K while the melting temperature found for the 18% case was 1580K. Although both of these are higher than the melting temperature of copper, they are fairly close. The reason for the melting temperature of the vane being higher than predicted is due to the presence of tungsten which has a melting temperature of 3683K.

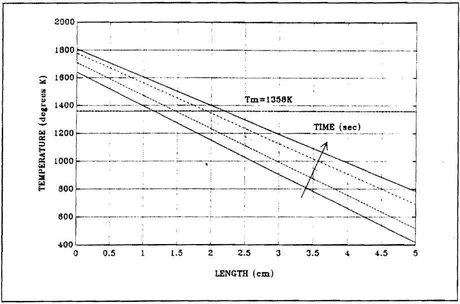


Figure 19 Temperature Profiles Between Nodes One and Two For the 9% Case

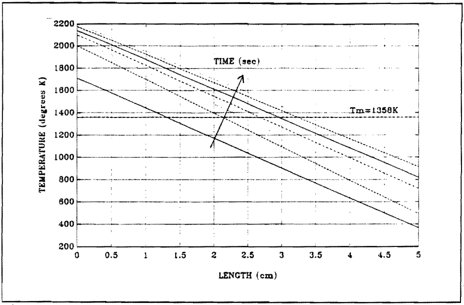


Figure 20 Temperature Profiles Between Nodes One and Two for the 18% Case

IV. DISCUSSION OF RESULTS

The full scale three node model attempted to show that the three fin nodes of the Parker five node full scale model could be reduced to one node. This was done, obtaining similar results for the convection heat transfer coefficients at the tip and fin. This validated the use of only one fin node in Reno's four node quarter scale model.

The erosion effects of aluminized propellent on the quarter scale vanes had to be investigated. There were three postulates considered of how the heat transfer coefficients changed:

(1) the heat transfer coefficients were independent of erosion rate and time,

(2) the heat transfer coefficients were dependent upon erosion rate, but given a fixed erosion rate, were time independent, and

(3) the heat transfer coefficients were dependent upon erosion rate and were time varient.

The first postulate was investigated by A. Danielson in [Ref. 2]. He found that as the percentage of aluminum in the exhaust and the erosion rate increased, nonlinear factors began to have a larger impact and show the limitations of the linear model [Ref. 2:p. 9].

To investigate the remaining two postulates, a model for the erosion of the vanes had to be developed. The erosion of the vanes was modeled as a linear decrease of the geometric dimensions as a function of time and mass loss percentage. This was done in the subprogram COEF.

For the second and third postulates, the coefficients in the PID subprogram COEF were set to the appropriate values for cases two and three, thereby allowing the geometric dimensions to vary. This led to excellent results which remained fairly constant even as the percentage of aluminum in the propellant increased. The sum of the squares error was only 1.19 for the 0% Al case, 0.73 for the 9% Al case, and 1.52 for the 18% Al case. This seemed to link the erosion rate to the heat transfer coefficients.

To determine whether the heat transfer coefficients were time dependent, the program COEF.FOR was modified to calculate the heat transfer coefficients as a function of time. Although the heat transfer coefficients remained fairly constant at the fin, they decreased over time at the tip.

An equation based on ablation of the vane was used to try to predict the erosion rate. The erosion rate was then integrated over the time of the motor firing to obtain the theoretical length of the vane which eroded. Although the 9% case predicted an eroded length which was more than double the experimental value, the 18% case was very close. Two of the reasons the 9% case was off can be explained by the simplicity

of the model and the assumption that ablation would being occuring at the melting temperature of copper instead of the tungsten-copper alloy which the vane was composed of.

To find a closer value to the melting temperature of the vane, the simulated vane temperature was plotted as a function of length between nodes one and two. By using the known length of vane eroded, a theoretical melting temperature could be found. The melting temperatures found were 1732K and 1580K for the 9% and 18% cases respectively. This was much closer to the 1358K for the melting temperature of copper than the 3513K for the tungsten-copper alloy of the vane.

CONCLUSIONS

- The five node full scale model can be reduced to a three node full scale model by removing two of the three fin nodes and produce comparable convective heat transfer coefficients.
- Erosion of thrust vector control vanes can be adequately modeled by a linear decrease of the geometric properties as a function of time and the percentage of aluminum used in the propellant.
- The negative values found for R_{4G} indicate heating of the vane from the mount area.
- Both the tip and fin convective heat transfer coefficients were dependant upon erosion rate and were time variant.
- The erosion rate and therefore the length of the vane which will erode over the time of a motor firing can be adequately predicted using an energy balance equation based upon ablation of the vane.
- The melting temperature of the vane appears to be much closer to that of copper than the tungsten-copper alloy which is expected.

RECOMMENDATIONS FOR FURTHER STUDY

- The erosion front modeling needs to be investigated further to see if erosion mechanisms other than ablation can be modeled such as direct impingement of the aluminized particles on the vane.
- The G-law erosion algorithm explained in [Ref. 9] may provide a method to use results from a quarter scale model to predict full scale heat transfer characteristics.
- The quarter scale model needs to be modified to include the heating effects in the vane mount area.

APPENDIX A. SIMULATION PROGRAM

This appendix contains the FORTRAN code used in the program SIM.FOR, which is a forward model program to simulate the temperatures of a three node full scale model, and SIM4.FOR which is a forward model program to simulate the temperatures of the four node quarter scale models.

```
c----------------------------------------------------------------

      Program SIM

c     This is a forward model program to simulate the
c     temperatures of a three node full scale model.

      integer maxparam,neq

      parameter (maxparam=50,neq=3)

      integer id0, istep, nout

      real*8 t,tend,a(3,3),b(3,3),u(3),t2(61),y(3)
      real*8 param(maxparam),fcn,float,a3g

      intrinsic float

      external fcn, divprk, sset

      common/data1/a,b,u

c     Open files for data input/output

      open(9,name='sim3.mat',  status='new')
      open(8,name='datam.dat', status='old')

c     read in experimental data

      do i=1,61
         read(8,*) t2(i)
      enddo
      close(8)

c     initialize matrices

      do i=1,3
          do j=1,3
              a(i,j)=0.0
              b(i,j)=0.0
          enddo
      enddo

c     enter data for trial run

      a(1,2)=0.2936
      a(2,1)=0.0147
      a(2,3)=0.0107
      a(3,2)=0.0243
```

```
      a3g=0.0001
      b(1,1)=1.0000
      b(2,2)=0.0500

      a(1,1)=-(a(1,2)+b(1,1))
      a(2,2)=-(a(2,1)+a(2,3)+b(2,2))
      a(3,3)=-(a(3,2)+a3g)

      u(1)=2670
      u(2)=2570
      u(3)=0.0

c     set initial conditions

      t=0.0
      do i=1,3
         y(i)=0.0
      enddo

      tol=0.0005
      call sset(maxparam, 0.0, param, 1)

      id0=1
      do istep=1,61
         tend=0.0768*float(istep)
         call DIVPRK(id0,neq,fcn,t,tend,tol,param,y)
         write(9,9001) t,t2(istep),y
      enddo

c     final call to release workspace

      id0=3
      call DIVPRK(id0,neq,fcn,t,tend,tol,param,y)

 9001    format(1f6.3,4f10.4)

      close(9)

      end
c------------------------------------------------------------

      subroutine fcn(neq,t,y,yprime)

      integer neq

      real*8 t,y(neq),yprime(neq)
      real*8 a(3,3),b(3,3),u(3),d

      common/data1/a,b,u
```

```
c    thrust profile simulation as step input

     if (t.gt.0.2) then
        d=1.0
     else
        d=0.0
     end if

     do i=1,neq
     yprime(i)=0.0
        do j=1,neq
        yprime(i)=yprime(i)+a(i,j)*y(j)+b(i,j)*u(j)*d
        enddo
     enddo

     return
     end
```

```
c--------------------------------------------------------

      Program SIM4

c     This is a forward model program to simulate the
      temperatures of a four node quarter scale model.

      integer maxparam,neq

      parameter (maxparam=50,neq=4)

      integer id0, istep, nout

      real*8 t,tend,a(4,4),b(4,4),u(4),y(4)
      real*8 param(maxparam),fcn,float,a4g

      intrinsic float

      external fcn, divprk, sset, coef

      common/data1/a,b,u

c     Open files for data input/output

      open(9,name='sim49.mat',  status='new')

c     initialize matrices

      do i=1,4
         do j=1,4
            a(i,j)=0.0
            b(i,j)=0.0
         enddo
      enddo

      u(1)=2155
      u(2)=2061
      u(3)=0.0
      u(4)=0.0

c     set initial conditions

      t=0.0
      do i=1,4
         y(i)=0.0
      enddo

      tol=0.0005
      call sset(maxparam, 0.0, param, 1)
```

```
      id0=1
      do istep=1,41
         tend=0.05*float(istep)
         call coef(tend)
         call DIVPRK(id0,neq,fcn,t,tend,tol,param,y)
         write(9,9001) t,y
      enddo

c     final call to release workspace

      id0=3
      call DIVPRK(id0,neq,fcn,t,tend,tol,param,y)

 9001     format(1f6.3,4f10.4)

      close(9)

      end

C---------------------------------------------------------------

      subroutine fcn(neq,t,y,yprime)

      integer neq

      real*8 t,y(neq),yprime(neq)
      real*8 a(4,4),b(4,4),u(4),d

      common/data1/a,b,u

c     thrust profile simulation as step input

      if (t.gt.0.7) then
         d=1.0
      else
         d=0.0
      end if

      do i=1,neq
      yprime(i)=0.0
         do j=1,neq
         yprime(i)=yprime(i)+a(i,j)*y(j)+b(i,j)*u(j)*d
         enddo
      enddo

      return
      end
C---------------------------------------------------------------

      subroutine coef(tend)
```

```
real*8 vt,vf,vs,atf,afs,ltf,lfs,rho,cp,k,sf
real*8 vt0,vf0,vs0,atf0,afs0,ltf0,lfs0,a12,a21,a23,a32
real*8 a(4,4),b(4,4),u(4),c1,c2,c3,r12,r23

common/data1/a,b,u

vt0=2.6
vf0=52.0
vs0=23.0

atf0=5.9
afs0=5.2

ltf0=5.0
lfs0=6.0

rho=18310.0
cp=146.0
k=173.0
sf=0.25

vt=vt0-0.0*tend
vf=vf0-0.0*tend
vs=vs0-0.0*tend

atf=atf0-0.0*tend
afs=afs0-0.0*tend

ltf=ltf0-0.0*tend
lfs=lfs0-0.0*tend

r12=100.0*ltf/(k*atf)
r23=100.0*lfs/(k*afs)
c1=rho*cp*vt*0.000001
c2=rho*cp*vf*0.000001
c3=rho*cp*vs*0.000001

r12=r12/sf
r23=r23/sf
c1=c1*sf**3
c2=c2*sf**3
c3=c3*sf**3

a(1,2)=1/(c1*r12)
a(2,1)=1/(c2*r12)
a(2,3)=1/(c2*r23)
a(3,2)=1/(c3*r23)

a(3,4)=0.5376
a(4,3)=0.1528
a4g=-0.1651
```

```
      b(1,1)=7.6511
      b(2,2)=0.0722

      a(1,1)=-(a(1,2)+b(1,1))
      a(2,2)=-(a(2,1)+a(2,3)+b(2,2))
      a(3,3)=-(a(3,2)+a(3,4))
      a(4,4)=-(a(4,3)+a4g)

      return
      end
c-----------------------------------------------------------
```

APPENDIX B. COEFFICIENT PROGRAM

This appendix contains the FORTRAN code used in the program COEF.FOR which calculated the effect of erosion on the known coefficients of the A matrix and the heat transfer coefficients.

```
c--------------------------------------------------------------
      program coef

      integer i

      real*8 vt,vf,vs,atf,afs,ltf,lfs,t,rho,cp,k,sf
      real*8 vt0,vf0,vs0,atf0,afs0,ltf0,lfs0,a12,a21,a23,a32
      real*8 asf0,asf,ast0,ast,b11,b22,ht,hf

      intrinsic float

      open(10,name='coef18.mat',status='new')
      open(11,name='htc18.mat',status='new')

      vt0=2.6
      vf0=52.0
      vs0=23.0

      atf0=5.9
      afs0=5.2

      ast0=4.35
      asf0=112.16

      ltf0=5.0
      lfs0=6.0

      rho=18310.0
      cp=146.0
      k=173.0
      sf=0.25

      b11=1.6236
      b22=0.05

      do i=1,33

         t=0.05*float(i)

         vt=vt0-0.8125*t
         vf=vf0-16.25*t
         vs=vs0-7.1875*t

         atf=atf0-0.0*t
         afs=afs0-0.0*t

         ast=ast0-0.90625*t
         asf=asf0-23.367*t

         ltf=ltf0-1.5625*t
```

```fortran
      lfs=lfs0-1.875*t

      r12=100.0*ltf/(k*atf)
      r23=100.0*lfs/(k*afs)
      c1=rho*cp*vt*0.000001
      c2=rho*cp*vf*0.000001
      c3=rho*cp*vs*0.000001

      r12=r12/sf
      r23=r23/sf
      c1=c1*sf**3
      c2=c2*sf**3
      c3=c3*sf**3
      ast=ast*sf**2
      asf=asf*sf**2

      a12=1/(c1*r12)
      a21=1/(c2*r12)
      a23=1/(c2*r23)
      a32=1/(c3*r23)

      rf1=1/(b11*c1)
      rf2=1/(b22*c2)

      ht=10000.0/(rf1*ast)
      hf=10000.0/(rf2*asf)

9999   format(1f10.4,2f10.2)
9998   format(5f10.5)

   write(10,9998)t,a12,a21,a23,a32
   write(11,9999)t,ht,hf
   end do
   close(10)
   close(11)

   end
```

APPENDIX C. PID PROGRAMS

This appendix contains the PID programs for the three node full scale model (NODE3), and the four node quarter scale models for propellant with 0% Al (NODE40), 9% Al (NODE49) and 18% Al (NODE418).

```
c------------------------------------------------------------

      Program NODE3

c     This program is the PID program for the three node vane
c     model.

      external temp

      integer m,n,iparm(6),ibtype,ldfjac

      parameter (m=61,n=3,ldfjac=m)

      real*8 rparm(7),x(n),f(m),xjac(m,n),xg(n),ssq,ub1,ub2
      real*8 xlb(n),xub(n),xscale(n),fscale(m),float,ht,hf
      real*8 a(3,3),b(3,3),u(3),t2(61),ys(3,61)
      real*8 rho,k,cp,sf,c1,c2,c3,r12,r23,a3g
      real*8 vt,vf,vs,atf,afs,ast,asf,ltf,lfs

c     Variables
c         m           = number of functions
c         n           = number of variables
c         iparm       = list of parameters for DBCLSF setup
c         ibtype      = type of bounds on variables
c         ldfjac      = leading dimension of fjac
c         rparm       = list of parameters for DBCLSF setup
c         x(n)        = the pt where the function is evaluated
c         f(m)        = the computed function at the point x
c         xjac(m,n)   = matrix containing a finite difference
c                       approx Jacobian at the approx solution
c         xg(n)       = initial guess of x
c         xlb(n)      = x lower bound
c         xub(n)      = x upper bound
c         xscale(n)   = vector containing the scaling matrix for
c                       the variables
c         fscale(m)   = vector containing the scaling matrix for
c                       the functions
c         ssq         = sum of the squares
c         a(3,3)      = a matrix
c         b(3,3)      = b matrix
c         u(3)        = [TR1, TR2, 0]
c         t2(61)      = experimental temperatures
c         ys(3,61)    = calculated temperatures
c         rho         = density
c         k           = conduction heat transfer coefficient
c         cp          = specific heat
c         vt          = volume of the tip
c         vf          = volume of the fin
c         vs          = volume of the shaft
c         atf         = cross sectional area from tip to fin
```

```
c       afs         = cross sectional area from fin to shaft
c       ast         = surface area of the tip
c       asf         = surface area of the fin
c       ltf         = length from tip to fin
c       lfs         = length from fin to shaft
c       sf          = scale factor
c       ub1         = stagnation temperature, TR1
c       ub2         = free stream temperature, TR2
c       ht          = convection heat transfer coefficient at
                      tip
c       hf          = convection heat transfer coefficient at
                      fin

      intrinsic float

      common/data1/a,b,u,t2,ys

c     Open files for data input/output

      open(10,name='result.dat', status='new')
      open(9,name='temp.mat',   status='new')
      open(8,name='datam.dat', status='old')
      open(7,name='input.dat', status='old')

c     read in experimental data

      do i=1,61
         read(8,*) t2(i)
      enddo
      close(8)

c     read in input data

      read(7,*)
      read(7,*)
      read(7,*)
      read(7,*)
      read(7,*) rho,k,cp
      read(7,*)
      read(7,*)
      read(7,*) vt, vf, vs
      read(7,*)
      read(7,*)
      read(7,*) atf, afs
      read(7,*)
      read(7,*)
      read(7,*) ast, asf
      read(7,*)
      read(7,*)
      read(7,*) ltf, lfs
      read(7,*)
```

```
      read(7,*)
      read(7,*) sf, ub1, ub2
      close(7)

c     initial conditions

c     full scale data

      r12=100.0*ltf/(k*atf)
      r23=100.0*lfs/(k*afs)
      c1=rho*cp*vt*0.000001
      c2=rho*cp*vf*0.000001
      c3=rho*cp*vs*0.000001

c     scaled data

      r12=r12/sf
      r23=r23/sf
      c1=c1*sf**3
      c2=c2*sf**3
      c3=c3*sf**3

c     initialize matrices to zero

      do i=1,3
      u(i)=0.0
         do j=1,3
         a(i,j)=0.0
         b(i,j)=0.0
         enddo
      enddo

      a(1,2)=1/(c1*r12)
      a(2,1)=1/(c2*r12)
      a(2,3)=1/(c2*r23)
      a(3,2)=1/(c3*r23)

      a3g=0.0
      b(1,1)=0.0
      b(2,2)=0.0

      a(1,1)=-(a(1,2)+b(1,1))
      a(2,2)=-(a(2,1)+a(2,3)+b(2,2))
      a(3,3)=-(a(3,2)+a3g)

      u(1)=ub1
      u(2)=ub2

      xg(1)=a3g
      xg(2)=b(1,1)
      xg(3)=b(2,2)
```

```
c   set up parameters for DBCLSF call

    do i=1,n
        xscale(i)=1.0
        xlb(i)=0.0001
        xub(i)=100.0
        xg(i)=0.01
        x(i)=0.0
    end do

    do i=1,m
        fscale(i)=1.0
    end do

    ibtype=0

    call dbclsf(temp,m,n,xg,ibtype,xlb,xub,xscale,fscale,
     &          iparm,rparm,x,f,xjac,ldfjac)

c   calculate unknown resistances and convection heat transfer
c   coefficients

    a3g   =x(1)
    b(1,1)=x(2)
    b(2,2)=x(3)

    c1=rho*cp*vt*0.000001
    c2=rho*cp*vf*0.000001
    c3=rho*cp*vs*0.000001

    c1=c1*sf**3
    c2=c2*sf**3
    c3=c3*sf**3

    rf1 =1/(b(1,1)*c1)
    rf2 =1/(b(2,2)*c2)
    r3g =1/(a3g*c3)

    ht =10000.0/(rf1*ast)
    hf =10000.0/(rf2*asf)

c   print and save results

    write(6,*) '        a3g             b11           b22'
    write(6,9000) x(1),x(2),x(3)

 9000   format(3f12.4)
 9003    format(2f12.4)

    write(10,*)'       a3g          b(1,1)       b(2,2)'
    write(10,9000) x(1),x(2),x(3)
```

```
      write(10,*)
      write(10,*)
      write(10,*)'       rf1          rf2        r3g'
      write(10,9000) rf1,rf2,r3g
      write(10,*)
      write(10,*)
      write(10,*)'        ht          hf'
      write(10,9003) ht,hf

c     write the temp-time data for MATLAB analysis

      do i=1,61
         tt=0.0768*float(i)
         write(9,9001)tt,ys(2,i),t2(i)
      enddo
 9001    format(1f6.2,2f10.3)

      close(10)
      close(9)

      end

c-----------------------------------------------------------------

      Subroutine TEMP (m,n,x,f)

c     This calculates the temperature-time history using the
c     current parameters supplied by DBCLSF called from PID. It
c     calculates an error function returned to DBCLSF based on
c     the differences between predicted and observed temperature
c     histories.

      integer maxparam, neq

      parameter(maxparam=50, neq=3)

      integer id0,istep,nout,m,n

      real*8 t,tend,y(3),tol,fcn,float,param(maxparam),
      real*8 x(3),f(61),coef
      real*8 a(3,3),b(3,3),u(3),t2(61),ys(3,61)
      real*8 rho,k,cp,sf,c1,c2,c3,r12,r23,a3g
      real*8 vt,vf,vs,atf,afs,ast,asf,ltf,lfs

      intrinsic float

      external fcn,divprk,sset

      common/data1/a,b,u,t2,ys

      open(12,name='incoming.dat',status='new')
```

```
      a3g   =x(1)
      b(1,1)=x(2)
      b(2,2)=x(3)

      write(6,8000)a3g,b(1,1),b(2,2)

 8000     format(3f12.4)

      a(1,1)=-(a(1,2)+b(1,1))
      a(2,2)=-(a(2,1)+a(2,3)+b(2,2))
      a(3,3)=-(a(3,2)+a3g)

c     Set initial conditions

      t=0.0
      do i=1,neq
      y(i)=0.0
         do j=1,61
         ys(i,j)=0.0
         enddo
      enddo
      tol=0.0005

      call sset (maxparm, 0.0, param, 1)

      id0=1
      do istep=1,61
         tend=0.0768*float(istep)
         CALL DIVPRK (id0, neq, fcn, t, tend, tol, param, y)
         do i=1,3
         ys(i,istep)=y(i)
         enddo
      enddo

c     Final call to release workspace

      id0=3
      call divprk (id0,neq,fcn,t,tend,tol,param,y)

c     calculate error functions

      do i=1,61
      f(i)=ys(2,i)-t2(i)
      enddo

c     print out rms error
      ssqr=0.0
      do i=1,61
      ssqr=ssqr+f(i)*f(i)
      enddo
      ssqr=ssqr/61
```

```
      xer=sqrt(ssqr)
      write(6,*) xer
      write(12,*) xer
      return
      end

C------------------------------------------------------------

      subroutine fcn(neq,t,y,yprime)

      integer neq

      real*8 t,y(neq),yprime(neq)
      real*8 a(3,3),b(3,3),u(3),d,ys(3,61)

      common/data1/a,b,u,t2,ys

c     thrust profile simulation as step input

      if (t.gt.0.2) then
         d=1.0
      else
         d=0.0
      end if

      do i=1,neq
      yprime(i)=0.0
         do j=1,neq
         yprime(i)=yprime(i)+a(i,j)*y(j)+b(i,j)*u(j)*d
         enddo
      enddo

      return
      end
```

```
c------------------------------------------------------------

      Program NODE40

c     This program is the PID program for the four node vane
c     model with ablation from exhaust with 0% Al.

      external temp

      integer m,n,iparm(6),ibtype,ldfjac

      parameter (m=122,n=5,ldfjac=m)

      real*8 rparm(7),x(n),f(m),xjac(m,n),xg(n),ssq,ub1,ub2
      real*8 xlb(n),xub(n),xscale(n),fscale(m),float,ht,hf
      real*8 a(4,4),b(4,4),u(4),t3(61),t4(61),ys(4,61)
      real*8 rho,k,cp,sf,c1,c2,c3,c4,r12,r23,a4g
      real*8 vt0,vf0,vs0,atf0,afs0,ast0,asf0,ltf0,lfs0
      real*8 vt,vf,vs,atf,afs,ast,asf,ltf,lfs

c     Variables
c         m             = number of functions
c         n             = number of variables
c         iparm         = list of parameters for DBCLSF setup
c         ibtype        = type of bounds on variables
c         ldfjac        = leading dimension of fjac
c         rparm         = list of parameters for DBCLSF setup
c         x(n)          = the pt where the function is evaluated
c         f(m)          = the computed function at the point x
c         xjac(m,n)     = matrix containing a finite difference
c                         approx Jacobian at the approx solution
c         xg(n)         = initial guess of x
c         xlb(n)        = x lower bound
c         xub(n)        = x upper bound
c         xscale(n)     = vector containing the scaling matrix for
c                         the variables
c         fscale(m)     = vector containing the scaling matrix for
c                         the functions
c         ssq           = sum of the squares
c         a(neq,neq)    = a matrix
c         b(neq,neq)    = b matrix
c         u(neq)        = [TR1, TR2, 0,   0]
c         t3(61)        = experimental temperatures at node 3
c         t4(61)        = experimental temperatures at node 4
c         ys(neq,61)    = calculated temperatures
c         rho           = density
c         k             = conduction heat transfer coefficient
c         cp            = specific heat
c         vt            = volume of the tip
c         vf            = volume of the fin
```

```
c       vs            = volume of the shaft
c       atf           = cross sectional area from tip to fin
c       afs           = cross sectional area from fin to shaft
c       ast           = surface area of the tip
c       asf           = surface area of the fin
c       ltf           = length from tip to fin
c       lfs           = length from fin to shaft
c       sf            = scale factor
c       ub1           = stagnation temperature, TR1
c       ub2           = free stream temperature, TR2
c       ht             = convection heat transfer coefficient at
                        tip
c       hf             = convection heat transfer coefficient at
                        fin

      intrinsic float

      common/data1/a,b,u,t3,t4,ys
      common/data2/rho,k,cp,sf,c1,c2,c3
      common/data3/vt0,vf0,vs0,atf0,afs0,ast0,asf0,ltf0,lfs0
      common/data4/vt,vf,vs,atf,afs,ast,asf,ltf,lfs

c     Open files for data input/output

      open(10,name='result0.dat', status='new')
      open(9,name='temp0.mat',  status='new')
      open(8,name='datam0.dat', status='old')
      open(7,name='input.dat', status='old')

c     read in experimental data

      do i=1,61
         read(8,*) t3(i)
      enddo
      do i=1,61
         read(8,*) t4(i)
      enddo
      close(8)

c     read in input data

      read(7,*)
      read(7,*)
      read(7,*)
      read(7,*)
      read(7,*) rho,k,cp
      read(7,*)
      read(7,*)
      read(7,*) vt0, vf0, vs0
      read(7,*)
      read(7,*)
```

```
      read(7,*) atf0, afs0
      read(7,*)
      read(7,*)
      read(7,*) ast0, asf0
      read(7,*)
      read(7,*)
      read(7,*) ltf0, lfs0
      read(7,*)
      read(7,*)
      read(7,*) sf, ub1, ub2
      close(7)

c     initial conditions

      t=0
      tend=0
      call coef(x,tend)

      u(1)=ub1
      u(2)=ub2
      u(3)=0.0
      u(4)=0.0

c     set up parameters for DBCLSF call

      do i=1,n
         xscale(i)=1.0
         xlb(i)=-0.2
         xub(i)=100.0
         xg(i)=0.1
         x(i)=0.0
      end do

      do i=1,m
         fscale(i)=1.0
      end do

      ibtype=0

      call dbclsf(temp,m,n,xg,ibtype,xlb,xub,xscale,fscale,
     &           iparm,rparm,x,f,xjac,ldfjac)

c     calculate unknown resistances and convection heat transfer
c     coefficients

      a(3,4)=x(1)
      a(4,3)=x(2)
      a4g   =x(3)
      b(1,1)=x(4)
      b(2,2)=x(5)
```

```
      c1=rho*cp*vt*0.000001
      c2=rho*cp*vf*0.000001
      c3=rho*cp*vs*0.000001

      c1=c1*sf**3
      c2=c2*sf**3
      c3=c3*sf**3

      rf1 =1/(b(1,1)*c1)
      rf2 =1/(b(2,2)*c2)
      r34 =1/(a(3,4)*c3)
      c4  =1/(a(4,3)*r34)
      r4g =1/(a4g*c4)

      ht =10000.0/(rf1*(ast*sf**2))
      hf =10000.0/(rf2*(asf*sf**2))

c     print and save results

      write(6,*) ' a34      a43      a4g         b11          b22'
      write(6,9000) x(1),x(2),x(3),x(4),x(5)

 9000   format(5f10.4)
 9003   format(2f11.4)

      write(10,*)'   a34         a43            a4g          b(1,1)
      b(2,2)'
      write(10,9000) x(1),x(2),x(3),x(4),x(5)
      write(10,*)
      write(10,*)
      write(10,*)'        rf1          rf2         r4g'
      write(10,9000) rf1,rf2,r4g
      write(10,*)
      write(10,*)
      write(10,*)'        ht           hf'
      write(10,9003) ht,hf

c     write the temp-time data for MATLAB analysis

      do i=1,61
          tt=0.05*float(i)
          write(9,9001)tt,ys(3,i),ys(4,i),t3(i),t4(i)
      enddo
 9001   format(2x,1f6.4,4f10.4)

      close(10)
      close(9)

      end

c-----------------------------------------------------------
```

```
      Subroutine TEMP (m,n,x,f)

c     This calculates the temperature-time history using the
c     current parameters supplied by DBCLSF called from PID. It
c     calculates an error function returned to DBCLSF based on
c     the differences between predicted and observed temperature
c     histories.

      integer maxparam, neq

      parameter(maxparam=50, neq=4)

      integer id0,istep,nout,m,n

      real*8 t,tend,y(4),tol,fcn,float,param(maxparam),
      real*8 x(n),f(m),coef
      real*8 a(4,4),b(4,4),u(4),t3(61),t4(61),ys(4,61)
      real*8 rho,k,cp,sf,c1,c2,c3,c4,r12,r23,a4g
      real*8 vt0,vf0,vs0,atf0,afs0,ast0,asf0,ltf0,lfs0
      real*8 vt,vf,vs,atf,afs,ast,asf,ltf,lfs

      intrinsic float

      external fcn,divprk,sset,coef

      common/data1/a,b,u,t3,t4,ys
      common/data2/rho,k,cp,sf,c1,c2,c3
      common/data3/vt0,vf0,vs0,atf0,afs0,ast0,asf0,ltf0,lfs0
      common/data4/vt,vf,vs,atf,afs,ast,asf,ltf,lfs

      open(12,name='incoming.dat',status='new')

      a(3,4)=x(1)
      a(4,3)=x(2)
      a4g   =x(3)
      b(1,1)=x(4)
      b(2,2)=x(5)

      write(6,8000)a(3,4),a(4,3),a4g,b(1,1),b(2,2)

 8000    format(5f10.4)

      a(1,1)=-(a(1,2)+b(1,1))
      a(2,2)=-(a(2,1)+a(2,3)+b(2,2))
      a(3,3)=-(a(3,2)+a(3,4))
      a(4,4)=-(a(4,3)+a4g)

c     Set initial conditions

      t=0.0
      do i=1,neq
```

```
    y(i)=0.0
        do j=1,61
        ys(i,j)=0.0
        enddo
    enddo

    tol=0.0005

    call sset (maxparm, 0.0, param, 1)

    id0=1
    do istep=1,61
        tend=0.05*float(istep)
        call coef(x,tend)
        CALL DIVPRK (id0, neq, fcn, t, tend, tol, param, y)
        do i=1,4
        ys(i,istep)=y(i)
        enddo
    enddo

c   Final call to release workspace

    id0=3
    call divprk (id0,neq,fcn,t,tend,tol,param,y)

c   calculate error functions

    do i=1,61
        f(i)=ys(3,i)-t3(i)
        f(i+61)=ys(4,i)-t4(i)
    enddo

c   print out rms error
    ssqr=0.0
    do i=1,m
    ssqr=ssqr+f(i)*f(i)
    enddo
    ssqr=ssqr/m
    xer=sqrt(ssqr)
    write(6,*) xer
    write(12,*) xer
    return
    end

C-------------------------------------------------------

    subroutine fcn(neq,t,y,yprime)

    integer neq

    real*8 t,y(neq),yprime(neq)
```

```
      real*8 a(4,4),b(4,4),u(4),d,ys(4,61)

      common/data1/a,b,u,t3,t4,ys
      common/data2/rho,k,cp,sf,c1,c2,c3
      common/data3/vt0,vf0,vs0,atf0,afs0,ast0,asf0,ltf0,lfs0
      common/data4/vt,vf,vs,atf,afs,ast,asf,ltf,lfs

c     thrust profile simulation as step input

      if (t.gt.0.3) then
         d=1.0
      else
         d=0.0
      end if

      do i=1,neq
      yprime(i)=0.0
         do j=1,neq
         yprime(i)=yprime(i)+a(i,j)*y(j)+b(i,j)*u(j)*d
         enddo
      enddo

      return
      end
C----------------------------------------------------------------

      subroutine coef(x,tend)

      integer i,j

      real*8 tend,x(5)
      real*8 a(4,4),b(4,4),u(4)
      real*8 rho,k,cp,sf,c1,c2,c3,c4,r12,r23,a4g
      real*8 vt0,vf0,vs0,atf0,afs0,ast0,asf0,ltf0,lfs0
      real*8 vt,vf,vs,atf,afs,ast,asf,ltf,lfs

      common/data1/a,b,u,t3,t4,ys
      common/data2/rho,k,cp,sf,c1,c2,c3
      common/data3/vt0,vf0,vs0,atf0,afs0,ast0,asf0,ltf0,lfs0
      common/data4/vt,vf,vs,atf,afs,ast,asf,ltf,lfs

c     a,b matrix modification due to ablation effects

c     full scale data

      vt=vt0-0.013*tend
      vf=vf0-0.26*tend
      vs=vs0-0.115*tend

      atf=atf0-0.0*tend
```

```
      afs=afs0-0.0*tend

      ast=ast0-0.0145*tend
      asf=asf0-0.374*tend

      ltf=ltf0-0.025*tend
      lfs=lfs0-0.03*tend

      r12=100.0*ltf/(k*atf)
      r23=100.0*lfs/(k*afs)
      c1=rho*cp*vt*0.000001
      c2=rho*cp*vf*0.000001
      c3=rho*cp*vs*0.000001

c     scaled data

      r12=r12/sf
      r23=r23/sf
      c1=c1*sf**3
      c2=c2*sf**3
      c3=c3*sf**3

      a(1,2)=1/(c1*r12)
      a(2,1)=1/(c2*r12)
      a(2,3)=1/(c2*r23)
      a(3,2)=1/(c3*r23)


      a(3,4)=x(1)
      a(4,3)=x(2)
      a4g   =x(3)
      b(1,1)=x(4)
      b(2,2)=x(5)

      a(1,1)=-(a(1,2)+b(1,1))
      a(2,2)=-(a(2,1)+a(2,3)+b(2,2))
      a(3,3)=-(a(3,2)+a(3,4))
      a(4,4)=-(a(4,3)+a4g)

      return
      end

c-----------------------------------------------------------
```

```
      Program NODE49

c     This program is the PID program for the four node vane
      model with erosion from exhaust with 9% Al.

      external temp
      integer m,n,iparm(6),ibtype,ldfjac

      parameter (m=82,n=5,ldfjac=m)

      real*8 rparm(7),x(n),f(m),xjac(m,n),xg(n),ssq,ub1,ub2
      real*8 xlb(n),xub(n),xscale(n),fscale(m),float,ht,hf
      real*8 a(4,4),b(4,4),u(4),t3(41),t4(41),ys(4,41)
      real*8 rho,k,cp,sf,c1,c2,c3,c4,r12,r23,a4g
      real*8 vt0,vf0,vs0,atf0,afs0,ast0,asf0,ltf0,lfs0
      real*8 vt,vf,vs,atf,afs,ast,asf,ltf,lfs

c     Variables
c         m           = number of functions
c         n           = number of variables
c         iparm       = list of parameters for DBCLSF setup
c         ibtype      = type of bounds on variables
c         ldfjac      = leading dimension of fjac
c         rparm       = list of parameters for DBCLSF setup
c         x(n)        = the pt where the function is evaluated
c         f(m)        = the computed function at the point x
c         xjac(m,n)   = matrix containing a finite difference
c                       approx Jacobian at the approx solution
c         xg(n)       = initial guess of x
c         xlb(n)      = x lower bound
c         xub(n)      = x upper bound
c         xscale(n)  = vector containing the scaling matrix for
c                 the variables
c         fscale(m)  = vector containing the scaling matrix for
c                 the functions
c         ssq         = sum of the squares
c         a(neq,neq) = a matrix
c         b(neq,neq) = b matrix
c         u(neq)      = [TR1, TR2, 0,  0]
c         t3(41)      = experimental temperatures at node 3
c         t4(41)      = experimental temperatures at node 4
c         ys(neq,41) = calculated temperatures
c         rho         = density
c         k           = conduction heat transfer coefficient
c         cp          = specific heat
c         vt          = volume of the tip
c         vf          = volume of the fin
c         vs          = volume of the shaft
c         atf         = cross sectional area from tip to fin
c         afs         = cross sectional area from fin to shaft
```

```
c       ast         = surface area of the tip
c       asf         = surface area of the fin
c       ltf         = length from tip to fin
c       lfs         = length from fin to shaft
c       sf          = scale factor
c       ub1         = stagnation temperature, TR1
c       ub2         = free stream temperature, TR2
c       ht          = convection heat transfer coefficient at
                      tip
c       hf          = convection heat transfer coefficient at
                      fin

    intrinsic float

    common/data1/a,b,u,t3,t4,ys
    common/data2/rho,k,cp,sf,c1,c2,c3
    common/data3/vt0,vf0,vs0,atf0,afs0,ast0,asf0,ltf0,lfs0
    common/data4/vt,vf,vs,atf,afs,ast,asf,ltf,lfs

c   Open files for data input/output

    open(10,name='result9.dat', status='new')
    open(9,name='temp9.mat',  status='new')
    open(8,name='datam9.dat', status='old')
    open(7,name='input.dat', status='old')

c   read in experimental data

    do i=1,41
       read(8,*) t3(i)
    enddo
    do i=1,41
       read(8,*) t4(i)
    enddo
    close(8)

c   read in input data

    read(7,*)
    read(7,*)
    read(7,*)
    read(7,*)
    read(7,*) rho,k,cp
    read(7,*)
    read(7,*)
    read(7,*) vt0, vf0, vs0
    read(7,*)
    read(7,*)
    read(7,*) atf0, afs0
    read(7,*)
    read(7,*)
```

```
      read(7,*) ast0, asf0
      read(7,*)
      read(7,*)
      read(7,*) ltf0, lfs0
      read(7,*)
      read(7,*)
      read(7,*) sf, ub1, ub2
      close(7)

c     initial conditions

      t=0
      tend=0
      call coef(x,tend)

      u(1)=ub1
      u(2)=ub2
      u(3)=0.0
      u(4)=0.0

c     set up parameters for DBCLSF call

      do i=1,n
          xscale(i)=1.0
          xlb(i)=-0.2
          xub(i)=100.0
          xg(i)=0.1
          x(i)=0.0
      end do

      do i=1,m
          fscale(i)=1.0
      end do

      ibtype=0

      call dbclsf(temp,m,n,xg,ibtype,xlb,xub,xscale,fscale,
     &         iparm,rparm,x,f,xjac,ldfjac)

c     calculate unknown resistances and convection heat transfer
c     coefficients

      a(3,4)=x(1)
      a(4,3)=x(2)
      a4g   =x(3)
      b(1,1)=x(4)
      b(2,2)=x(5)

      c1=rho*cp*vt*0.000001
      c2=rho*cp*vf*0.000001
      c3=rho*cp*vs*0.000001
```

```
      c1=c1*sf**3
      c2=c2*sf**3
      c3=c3*sf**3

      rf1 =1/(b(1,1)*c1)
      rf2 =1/(b(2,2)*c2)
      r34 =1/(a(3,4)*c3)
      c4  =1/(a(4,3)*r34)
      r4g =1/(a4g*c4)

      ht =10000.0/(rf1*(ast*sf**2))
      hf =10000.0/(rf2*(asf*sf**2))

c     print and save results

      write(6,*) '  a34       a43       a4g        b11        b22'
      write(6,9000) x(1),x(2),x(3),x(4),x(5)

 9000   format(5f10.4)
 9003   format(5x,2f10.4)

      write(10,*)' a34      a43       a4g      b(1,1)     b(2,2)'
      write(10,9000) x(1),x(2),x(3),x(4),x(5)
      write(10,*)
      write(10,*)
      write(10,*)'          rf1          rf2        r4g'
      write(10,9000) rf1,rf2,r4g
      write(10,*)
      write(10,*)
      write(10,*)'        ht            hf'
      write(10,9003) ht,hf

c     write the temp-time data for MATLAB analysis

      do i=1,41
         tt=0.05*float(i)
         write(9,9001)tt,ys(3,i),ys(4,i),t3(i),t4(i)
      enddo
 9001   format(2x,1f6.4,4f10.4)

      close(10)
      close(9)

      end

c-----------------------------------------------------------------

      Subroutine TEMP (m,n,x,f)

c     This calculates the temperature-time history using the
c     current parameters supplied by DBCLSF called from PID. It
```

```
c   calculates an error function returned to DBCLSF based on
c   the differences between predicted and observed temperature
c   histories.

    integer maxparam, neq

    parameter(maxparam=50, neq=4)

    integer id0,istep,nout,m,n

    real*8t,tend,y(4),tol,fcn,float,param(50),x(n),f(m),coef
    real*8 a(4,4),b(4,4),u(4),t3(41),t4(41),ys(4,41)
    real*8 rho,k,cp,sf,c1,c2,c3,c4,r12,r23,a4g
    real*8 vt0,vf0,vs0,atf0,afs0,ast0,asf0,ltf0,lfs0
    real*8 vt,vf,vs,atf,afs,ast,asf,ltf,lfs

    intrinsic float

    external fcn,divprk,sset,coef

    common/data1/a,b,u,t3,t4,ys
    common/data2/rho,k,cp,sf,c1,c2,c3
    common/data3/vt0,vf0,vs0,atf0,afs0,ast0,asf0,ltf0,lfs0
    common/data4/vt,vf,vs,atf,afs,ast,asf,ltf,lfs

    open(12,name='incoming9.dat',status='new')

    a(3,4)=x(1)
    a(4,3)=x(2)
    a4g   =x(3)
    b(1,1)=x(4)
    b(2,2)=x(5)

    write(6,8000)a(3,4),a(4,3),a4g,b(1,1),b(2,2)
 8000    format(5f10.4)

    a(1,1)=-(a(1,2)+b(1,1))
    a(2,2)=-(a(2,1)+a(2,3)+b(2,2))
    a(3,3)=-(a(3,2)+a(3,4))
    a(4,4)=-(a(4,3)+a4g)

c   Set initial conditions

    t=0.0
    do i=1,neq
    y(i)=0.0
        do j=1,41
        ys(i,j)=0.0
        enddo
    enddo
```

```
      tol=0.0005

      call sset (maxparm, 0.0, param, 1)

      id0=1
      do istep=1,41
         tend=0.05*float(istep)
         call coef(x,tend)
         CALL DIVPRK (id0, neq, fcn, t, tend, tol, param, y)
         do i=1,4
         ys(i,istep)=y(i)
         enddo
      enddo

c     Final call to release workspace

      id0=3
      call divprk (id0,neq,fcn,t,tend,tol,param,y)

c     calculate error functions

      do i=1,41
         f(i)=ys(3,i)-t3(i)
         f(i+41)=ys(4,i)-t4(i)
      enddo

c     print out rms error
      ssqr=0.0
      do i=1,m
         ssqr=ssqr+f(i)*f(i)
      enddo
      ssqr=ssqr/m
      xer=sqrt(ssqr)
      write(6,*) xer
      write(12,*) xer
      return
      end

C-----------------------------------------------------------

      subroutine fcn(neq,t,y,yprime)

      integer neq

      real*8 t,y(neq),yprime(neq)
      real*8 a(4,4),b(4,4),u(4),d,ys(4,41)

      common/data1/a,b,u,t3,t4,ys
      common/data2/rho,k,cp,sf,c1,c2,c3
      common/data3/vt0,vf0,vs0,atf0,afs0,ast0,asf0,ltf0,lfs0
      common/data4/vt,vf,vs,atf,afs,ast,asf,ltf,lfs
```

```
c   thrust profile simulation as step input

    if (t.gt.0.7) then
       d=1.0
    else
       d=0.0
    end if

    do i=1,neq
    yprime(i)=0.0
       do j=1,neq
       yprime(i)=yprime(i)+a(i,j)*y(j)+b(i,j)*u(j)*d
       enddo
    enddo

    return
    end

C-----------------------------------------------------------

    subroutine coef(x,tend)

    integer i,j

    real*8 tend,x(5)
    real*8 a(4,4),b(4,4),u(4)
    real*8 rho,k,cp,sf,c1,c2,c3,c4,r12,r23,a4g
    real*8 vt0,vf0,vs0,atf0,afs0,ast0,asf0,ltf0,lfs0
    real*8 vt,vf,vs,atf,afs,ast,asf,ltf,lfs

    common/data1/a,b,u,t3,t4,ys
    common/data2/rho,k,cp,sf,c1,c2,c3
    common/data3/vt0,vf0,vs0,atf0,afs0,ast0,asf0,ltf0,lfs0
    common/data4/vt,vf,vs,atf,afs,ast,asf,ltf,lfs

c   a,b matrix modification due to ablation effects

c   full scale data

    vt=vt0-0.0*tend
    vf=vf0-0.0*tend
    vs=vs0-0.0*tend

    atf=atf0-0.0*tend
    afs=afs0-0.0*tend

    ast=ast0-0.0*tend
    asf=asf0-0.0*tend

    ltf=ltf0-0.0*tend
    lfs=lfs0-0.0*tend
```

```
      r12=100.0*ltf/(k*atf)
      r23=100.0*lfs/(k*afs)
      c1=rho*cp*vt*0.000001
      c2=rho*cp*vf*0.000001
      c3=rho*cp*vs*0.000001

c     scaled data

      r12=r12/sf
      r23=r23/sf
      c1=c1*sf**3
      c2=c2*sf**3
      c3=c3*sf**3

      a(1,2)=1/(c1*r12)
      a(2,1)=1/(c2*r12)
      a(2,3)=1/(c2*r23)
      a(3,2)=1/(c3*r23)

      a(3,4)=x(1)
      a(4,3)=x(2)
      a4g   =x(3)
      b(1,1)=x(4)
      b(2,2)=x(5)

      a(1,1)=-(a(1,2)+b(1,1))
      a(2,2)=-(a(2,1)+a(2,3)+b(2,2))
      a(3,3)=-(a(3,2)+a(3,4))
      a(4,4)=-(a(4,3)+a4g)

      return
      end

c-----------------------------------------------------------
```

```
      Program NODE418

c     This program is the PID program for the four node vane
      model with ablation from exhaust with 18% Al.

      external temp

      integer m,n,iparm(6),ibtype,ldfjac

      parameter (m=66,n=5,ldfjac=m)

      real*8 rparm(7),x(n),f(m),xjac(m,n),xg(n),ssq,ub1,ub2
      real*8 xlb(n),xub(n),xscale(n),fscale(m),float,ht,hf
      real*8 a(4,4),b(4,4),u(4),t3(33),t4(33),ys(4,33)
      real*8 rho,k,cp,sf,c1,c2,c3,c4,r12,r23,a4g
      real*8 vt0,vf0,vs0,atf0,afs0,ast0,asf0,ltf0,lfs0
      real*8 vt,vf,vs,atf,afs,ast,asf,ltf,lfs

c     Variables
c        m            = number of functions
c        n            = number of variables
c        iparm        = list of parameters for DBCLSF setup
c        ibtype       = type of bounds on variables
c        ldfjac       = leading dimension of fjac
c        rparm        = list of parameters for DBCLSF setup
c        x(n)         = the pt where the function is evaluated
c        f(m)         = the computed function at the point x
c        xjac(m,n)    = matrix containing a finite difference
c                       approx Jacobian at the approx solution
c        xg(n)        = initial guess of x
c        xlb(n)       = x lower bound
c        xub(n)       = x upper bound
c        xscale(n)  = vector containing the scaling matrix for
c                 the variables
c        fscale(m)  = vector containing the scaling matrix for
c                 the functions
c        ssq          = sum of the squares
c        a(neq,neq) = a matrix
c        b(neq,neq) = b matrix
c        u(neq)       = [TR1, TR2, 0,  0]
c        t3(33)       = experimental temperatures at node 3
c        t4(33)       = experimental temperatures at node 4
c        ys(neq,33) = calculated temperatures
c        rho          = density
c        k            = conduction heat transfer coefficient
c        cp           = specific heat
c        vt           = volume of the tip
c        vf           = volume of the fin
c        vs           = volume of the shaft
c        atf          = cross sectional area from tip to fin
```

```
c       afs         = cross sectional area from fin to shaft
c       ast         = surface area of the tip
c       asf         = surface area of the fin
c       ltf         = length from tip to fin
c       lfs         = length from fin to shaft
c       sf          = scale factor
c       ub1         = stagnation temperature, TR1
c       ub2         = free stream temperature, TR2
c       ht          = convection heat transfer coefficient at
                      tip
c       hf          = convection heat transfer coefficient at
                      fin

    intrinsic float

    common/data1/a,b,u,t3,t4,ys
    common/data2/rho,k,cp,sf,c1,c2,c3
    common/data3/vt0,vf0,vs0,atf0,afs0,ast0,asf0,ltf0,lfs0
    common/data4/vt,vf,vs,atf,afs,ast,asf,ltf,lfs

c   Open files for data input/output

    open(10,name='result18.dat', status='new')
    open(9,name='temp18.mat',  status='new')
    open(8,name='datam181.dat', status='old')
    open(7,name='input.dat', status='old')

c   read in experimental data

    do i=1,33
       read(8,*) t3(i)
    enddo
    do i=1,33
       read(8,*) t4(i)
    enddo
    close(8)

c   read in input data

    read(7,*)
    read(7,*)
    read(7,*)
    read(7,*)
    read(7,*) rho,k,cp
    read(7,*)
    read(7,*)
    read(7,*) vt0, vf0, vs0
    read(7,*)
    read(7,*)
    read(7,*) atf0, afs0
    read(7,*)
```

```
      read(7,*)
      read(7,*) ast0, asf0
      read(7,*)
      read(7,*)
      read(7,*) ltf0, lfs0
      read(7,*)
      read(7,*)
      read(7,*) sf, ub1, ub2
      close(7)

c     initial conditions

      t=0
      tend=0
      call coef(x,tend)

      u(1)=ub1
      u(2)=ub2
      u(3)=0.0
      u(4)=0.0

c     set up parameters for DBCLSF call

      do i=1,n
          xscale(i)=1.0
          xlb(i)=-0.2
          xub(i)=100.0
          xg(i)=0.1
          x(i)=0.0
      end do

      do i=1,m
          fscale(i)=1.0
      end do

      ibtype=0

      call dbclsf(temp,m,n,xg,ibtype,xlb,xub,xscale,fscale,
     &          iparm,rparm,x,f,xjac,ldfjac)

c     calculate unknown resistances and convection heat transfer
c     coefficients

      a(3,4)=x(1)
      a(4,3)=x(2)
      a4g   =x(3)
      b(1,1)=x(4)
      b(2,2)=x(5)

      c1=rho*cp*vt*0.000001
      c2=rho*cp*vf*0.000001
```

```
      c3=rho*cp*vs*0.000001

      c1=c1*sf**3
      c2=c2*sf**3
      c3=c3*sf**3

      rf1 =1/(b(1,1)*c1)
      rf2 =1/(b(2,2)*c2)
      r34 =1/(a(3,4)*c3)
      c4  =1/(a(4,3)*r34)
      r4g =1/(a4g*c4)

      ht =10000.0/(rf1*(ast*sf**2))
      hf =10000.0/(rf2*(asf*sf**2))

c     print and save results

      write(6,*) '  a34      a43       a4g       b11         b22'
      write(6,9000) x(1),x(2),x(3),x(4),x(5)

 9000  format(5f10.4)
 9003   format(2x,2f10.4)

      write(10,*)' a34      a43      a4g    b(1,1)      b(2,2)'
      write(10,9000) x(1),x(2),x(3),x(4),x(5)
      write(10,*)
      write(10,*)
      write(10,*)'        rf1          rf2        r4g'
      write(10,9000) rf1,rf2,r4g
      write(10,*)
      write(10,*)
      write(10,*)'        ht           hf'
      write(10,9003) ht,hf

c     write the temp-time data for MATLAB analysis

      do i=1,33
          tt=0.05*float(i)
          write(9,9001)tt,ys(3,i),ys(4,i),t3(i),t4(i)
      enddo
 9001   format(2x,1f6.4,4f10.4)

      close(10)
      close(9)

      end
c-------------------------------------------------------

      Subroutine TEMP (m,n,x,f)
```

```
c   This calculates the temperature-time history using the
c   current parameters supplied by DBCLSF called from PID. It
c   calculates an error function returned to DBCLSF based on
c   the differences between predicted and observed temperature
c   histories.

    integer maxparam, neq

    parameter(maxparam=50, neq=4)

    integer id0,istep,nout,m,n

    real*8 t,tend,y(4),tol,fcn,float,param(maxparam),
    real*8 x(n),f(m),coef
    real*8 a(4,4),b(4,4),u(4),t3(33),t4(33),ys(4,33)
    real*8 rho,k,cp,sf,c1,c2,c3,c4,r12,r23,a4g
    real*8 vt0,vf0,vs0,atf0,afs0,ast0,asf0,ltf0,lfs0
    real*8 vt,vf,vs,atf,afs,ast,asf,ltf,lfs

    intrinsic float

    external fcn,divprk,sset,coef

    common/data1/a,b,u,t3,t4,ys
    common/data2/rho,k,cp,sf,c1,c2,c3
    common/data3/vt0,vf0,vs0,atf0,afs0,ast0,asf0,ltf0,lfs0
    common/data4/vt,vf,vs,atf,afs,ast,asf,ltf,lfs

    open(12,name='incoming18.dat',status='new')

    a(3,4)=x(1)
    a(4,3)=x(2)
    a4g   =x(3)
    b(1,1)=x(4)
    b(2,2)=x(5)

    write(6,8000)a(3,4),a(4,3),a4g,b(1,1),b(2,2)

 8000   format(5f10.4)

    a(1,1)=-(a(1,2)+b(1,1))
    a(2,2)=-(a(2,1)+a(2,3)+b(2,2))
    a(3,3)=-(a(3,2)+a(3,4))
    a(4,4)=-(a(4,3)+a4g)

c   Set initial conditions

    t=0.0
    do i=1,neq
    y(i)=0.0
       do j=1,33
```

```
         ys(i,j)=0.0
         enddo
      enddo

      tol=0.0005

      call sset (maxparm, 0.0, param, 1)

      id0=1
      do istep=1,33
         tend=0.05*float(istep)
         call coef(x,tend)
         CALL DIVPRK (id0, neq, fcn, t, tend, tol, param, y)
         do i=1,4
         ys(i,istep)=y(i)
         enddo
      enddo

c     Final call to release workspace

      id0=3
      call divprk (id0,neq,fcn,t,tend,tol,param,y)

c     calculate error functions

      do i=1,33
      f(i)=ys(3,i)-t3(i)
      f(i+33)=ys(4,i)-t4(i)
      enddo

c     print out rms error
      ssqr=0.0
      do i=1,m
      ssqr=ssqr+f(i)*f(i)
      enddo
      ssqr=ssqr/m
      xer=sqrt(ssqr)
      write(6,*) xer
      write(12,*) xer
      return
      end

C--------------------------------------------------------------

      subroutine fcn(neq,t,y,yprime)

      integer neq

      real*8 t,y(neq),yprime(neq)
      real*8 a(4,4),b(4,4),u(4),d,ys(4,33)
```

```
      common/data1/a,b,u,t3,t4,ys
      common/data2/rho,k,cp,sf,c1,c2,c3
      common/data3/vt0,vf0,vs0,atf0,afs0,ast0,asf0,ltf0,lfs0
      common/data4/vt,vf,vs,atf,afs,ast,asf,ltf,lfs

c     thrust profile simulation as step input

      if (t.gt.0.1) then
         d=1.0
      else
         d=0.0
      end if

      do i=1,neq
      yprime(i)=0.0
         do j=1,neq
         yprime(i)=yprime(i)+a(i,j)*y(j)+b(i,j)*u(j)*d
         enddo
      enddo

      return
      end
C-------------------------------------------------------------

      subroutine coef(x,tend)

      integer i,j

      real*8 tend,x(5)
      real*8 a(4,4),b(4,4),u(4)
      real*8 rho,k,cp,sf,c1,c2,c3,c4,r12,r23,a4g
      real*8 vt0,vf0,vs0,atf0,afs0,ast0,asf0,ltf0,lfs0
      real*8 vt,vf,vs,atf,afs,ast,asf,ltf,lfs

      common/data1/a,b,u,t3,t4,ys
      common/data2/rho,k,cp,sf,c1,c2,c3
      common/data3/vt0,vf0,vs0,atf0,afs0,ast0,asf0,ltf0,lfs0
      common/data4/vt,vf,vs,atf,afs,ast,asf,ltf,lfs

c     a,b matrix modification due to ablation effects

c     full scale data

      vt=vt0-0.0*tend
      vf=vf0-0.0*tend
      vs=vs0-0.0*tend

      atf=atf0-0.0*tend
      afs=afs0-0.0*tend
```

```
      ast=ast0-0.0*tend
      asf=asf0-0.0*tend

      ltf=ltf0-0.0*tend
      lfs=lfs0-0.0*tend

      r12=100.0*ltf/(k*atf)
      r23=100.0*lfs/(k*afs)
      c1=rho*cp*vt*0.000001
      c2=rho*cp*vf*0.000001
      c3=rho*cp*vs*0.000001

c     scaled data

      r12=r12/sf
      r23=r23/sf
      c1=c1*sf**3
      c2=c2*sf**3
      c3=c3*sf**3

      a(1,2)=1/(c1*r12)
      a(2,1)=1/(c2*r12)
      a(2,3)=1/(c2*r23)
      a(3,2)=1/(c3*r23)

      a(3,4)=x(1)
      a(4,3)=x(2)
      a4g   =x(3)
      b(1,1)=x(4)
      b(2,2)=x(5)

      a(1,1)=-(a(1,2)+b(1,1))
      a(2,2)=-(a(2,1)+a(2,3)+b(2,2))
      a(3,3)=-(a(3,2)+a(3,4))
      a(4,4)=-(a(4,3)+a4g)

      return
      end

c----------------------------------------------------------------
```

APPENDIX D. PHYSICAL DATA FILES

The physical data files for the PID programs which contains the geometric and material properties of the vanes and the recovery temperatures used in each case.

NAWC INVERSE HEAT TRANSFER PROGRAM. INPUT DATA FOR NODE3.FOR

Material properties:

```
rho (kg/m^3),           k (w/m-deg k),          Cp (J/kg deg k)
   18310.0                 173.0                   146.0

Vol (tip, cm^3),        Vol (fin, cm^3),        Vol (shaft, cm^3)
    2.6                      52.00                   23.0

X-section areas:  tip-fin (cm^2)      fin-shaft (cm^2)
                      5.9                 5.2

Surface areas:     tip (cm^2)          fin (cm^2)
                      4.35           112.16

Conductive path     tip-fin (cm)      fin-shaft (cm)
                      5.0               6.0

Scale factor:      TR1          TR2
     1.00         2670         2570
```

NAWC INVERSE HEAT TRANSFER PROGRAM. INPUT DATA FOR NODE40.FOR

Material properties:

rho (kg/m^3),	k (w/m-deg k),	Cp (J/kg deg k)
18310.0	173.0	146.0

Vol (tip, cm^3),	Vol (fin, cm^3),	Vol (shaft, cm^3)
2.6	52.00	23.0

X-section areas:	tip-fin (cm^2)	fin-shaft (cm^2)
	5.9	5.2
Surface areas:	tip (cm^2)	fin (cm^2)
	4.35	112.16
Conductive path	tip-fin (cm)	fin-shaft (cm)
	5.0	6.0

Scale factor:	TR1	TR2
0.25	2360	2260

NAWC INVERSE HEAT TRANSFER PROGRAM. INPUT DATA FOR NODE49.FOR

Material properties:

rho (kg/m^3),	k (w/m-deg k),	Cp (J/kg deg k)
18310.0	173.0	146.0
Vol (tip, cm^3),	Vol (fin, cm^3),	Vol (shaft, cm^3)
2.6	52.00	23.0

X-section areas:	tip-fin (cm^2)	fin-shaft (cm^2)
	5.9	5.2
Surface areas:	tip (cm^2)	fin (cm^2)
	4.35	112.16
Conductive path	tip-fin (cm)	fin-shaft (cm)
	5.0	6.0

Scale factor:	TR1	TR2
0.25	2464	2370

NAWC INVERSE HEAT TRANSFER PROGRAM. INPUT DATA FOR NODE418.FOR

Material properties:

rho (kg/m^3),	k (w/m-deg k),	Cp (J/kg deg k)
18310.0	173.0	146.0
Vol (tip, cm^3),	Vol (fin, cm^3),	Vol (shaft, cm^3)
2.6	52.00	23.0

X-section areas:	tip-fin (cm^2)	fin-shaft (cm^2)
	5.9	5.2
Surface areas:	tip (cm^2)	fin (cm^2)
	4.35	112.16
Conductive path	tip-fin (cm)	fin-shaft (cm)
	5.0	6.0

Scale factor:	TR1	TR2
0.25	2970	2870

APPENDIX E. IMSL ROUTINES

A description of the IMSL routines DBCLSF, DIVPRK, and SSET used in the PID and simulation programs.

BCLSF/DBCLSF (Single/Double precision)

Purpose: Solve a nonlinear least squares problem subject to bounds on the variables using a modified Levenberg-Marquardt algorithm and a finite-difference Jacobian.

Usage:
```
CALL BCLSF (FCN, M, N, XGUESS, IBTYPE, XLB, XUB, XSCALE,
            FSCALE, IPARAM, RPARAM, X, FVEC, FJAC,
            LDFJAC)
```

Arguments

```
FCN    - User-supplied SUBROUTINE to evaluate the function to be
         minimized.  The usage is
         CALL FCN (M, N, X, F), where
         M      - Length of F.  (Input)
         N      - Length of X.  (Input)
         X      - The point at which the function is evaluated.
                  (Input)
                  X should not be changed by FCN.
         F      - The computed function at the point X.
                  (Output)
         FCN must be declared EXTERNAL in the calling program.
M      - Number of functions.  (Input)
N      - Number of variables.  (Input)
XGUESS - Vector of length N containing the initial guess.  (Input)
IBTYPE - Scalar indicating the types of bounds on variables.
         (Input)
         IBTYPE  Action
            0    User will supply all the bounds.
            1    All variables are nonnegative.
            2    All variables are nonpositive.
            3    User supplies only the bounds on 1st variable,
                 all other variables will have the same bounds.
XLB    - Vector of length N containing the lower bounds on
         variables.  (Input, if IBTYPE = 0; output, if IBTYPE = 1
         or 2; input/output, if IBTYPE = 3)
XUB    - Vector of length N containing the upper bounds on
         variables.  (Input, if IBTYPE = 0; output, if IBTYPE = 1
         or 2; input/output, if IBTYPE = 3)
XSCALE - Vector of length N containing the diagonal scaling matrix
         for the variables.  (Input)
         In the absence of other information, set all entries to
         1.0.
FSCALE - Vector of length M containing the diagonal scaling matrix
```

for the functions. (Input)
In the absence of other information, set all entries to 1.0.

IPARAM - Parameter vector of length 6. (Input/Output)
See Remarks.

RPARAM - Parameters vector of length 7. (Input/Output)
See Remarks.

X - Vector of length N containing the approximate solution. (Output)

FVEC - Vector of length M containing the residuals at the approximate solution. (Output)

FJAC - M by N matrix containing a finite difference approximate Jacobian at the approximate solution. (Output)

LDFJAC - Leading dimension of FJAC exactly as specified in the dimension statement of the calling program. (Input)

Remarks

1. Automatic workspace usage is

```
          BCLSF     14*N + 2*M - 1 units, or
          DBCLSF    26*N + 4*M - 2 units.
```

Workspace may be explicitly provided, if desired, by use of B2LSF/DB2LSF. The reference is

```
          CALL B2LSF (FCN, M, N, XGUESS, IBTYPE, XLB, XUB, XSCALE,
                      FSCALE, IPARAM, RPARAM, X, FVEC, FJAC,
                      LDFJAC, WK, IWK)
```

The additional arguments are as follows:

WK - Work vector of length 12*N + 2*M - 1. WK contains the following information on output:
The second N locations contain the last step taken.
The third N locations contain the last Gauss-Newton step.
The fourth N locations contain an estimate of the gradient at the solution.

IWK - Work vector of length 2*N containing the permutations used in the QR factorization of the Jacobian at the solution.

2. Informational errors

Type	Code	
3	1	Both the actual and predicted relative reductions in the function are less than or equal to the relative function convergence tolerance.
4	2	The iterates appear to be converging to a noncritical point.
4	3	Maximum number of iterations exceeded.

4 4 Maximum number of function evaluations exceeded.
4 5 Five consecutive steps have been taken with the maximum step length.

3. The first stopping criterion for BCLSF occurs when the norm of the function is less than the absolute function tolerance. The second stopping criterion occurs when the norm of the scaled gradient is less than the given gradient tolerance. The third stopping criterion for BCLSF occurs when the scaled distance between the last two steps is less than the step tolerance.

4. If nondefault parameters are desired for IPARAM or RPARAM, then U4LSF is called and the corresponding parameters are set to the desired value before calling the optimization program. Otherwise, if the default parameters are desired, then set IPARAM(1) to zero and call the optimization program omitting the call to U4LSF. The call to U4LSF would be as follows:

```
CALL U4LSF (IPARAM, RPARAM).
```

The following is a list of the parameters and the default values:

```
IPARAM - Integer vector of length 6.
         IPARAM(1) = Initialization flag. (0)
         IPARAM(2) = Number of good digits in the function.
                     (Machine dependent)
         IPARAM(3) = Maximum number of iterations. (100)
         IPARAM(4) = Maximum number of function evaluations. (400)
         IPARAM(5) = Maximum number of Jacobian evaluations. (100)
                     (Not used in BCLSF.)
         IPARAM(6) = Internal variable scaling flag.  (1)
                     If IPARAM(6) = 1 the values for XSCALE are
                     set internally.
RPARAM - Real vector of length 7.
         RPARAM(1) = Scaled gradient tolerance.
                     (SQRT(eps) in single precision)
                     (eps**(1/3) in double precision)
         RPARAM(2) = Scaled step tolerance. (eps**(2/3))
         RPARAM(3) = Relative function tolerance.
                     (MAX(1.0E-10,eps**(2/3)) in single precision)
                     (MAX(1.0D-20,eps**(2/3)) in double precision)
         RPARAM(4) = Absolute function tolerance.
                     (MAX(1.0E-20,eps**2) in single precision)
                     (MAX(1.0D-40,eps**2) in double precision)
         RPARAM(5) = False convergence tolerance. (100*eps)
         RPARAM(6) = Maximum allowable step size.
```

```
                        (1000*MAX(TOL1,TOL2)) where,
                        TOL1 = SQRT(sum of (XSCALE(I)*XGUESS(I))**2)
                               for I = 1,...,N
                        TOL2 = 2-norm of XSCALE.
          RPARAM(7) = Size of initial trust region radius.
                        (Based on the initial scaled Cauchy step)
   eps is machine epsilon.
   If double precision is desired, then DU4LSF is called and RPARAM
   is declared double precision.
```

Keywords: Levenberg-Marquardt; Trust region

Algorithm

BCLSF uses a modified Levenberg-Marquardt method and an active set strategy to solve nonlinear least squares problems subject to simple bounds on the variables. The problem is stated as follows:

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} F(x)^T F(x) = \frac{1}{2} \sum_{i=1}^{m} f_i(x)^2$$

subject to $l \leq x \leq u$.

where $m \geq n$. $F : \mathbb{R}^n \to \mathbb{R}^m$, and $f_i(x)$ is the i-th component function of $F(x)$. From a given starting point, an active set IA, which contains the indices of the variables at their bounds, is built. A variable is called a 'free variable' if it is not in the active set. The routine then computes the search direction for the free variables according to the formula

$$d = -(J^T J + \mu I)^{-1} J^T F,$$

where μ is the Levenberg-Marquardt parameter, $F = F(x)$, and J is the Jacobian with respect to the free variables. The search direction for the variables in IA is set to zero. The trust region approach discussed by Dennis and Schnabel (1983) is used to find the new point. Finally, the optimality conditions are checked. The conditions are:

$$\|g(x_i)\| \leq \epsilon, \quad l_i < x_i < u_i$$

$$g(x_i) < 0, \quad x_i = u_i$$

$$g(x_i) > 0, \quad x_i = l_i.$$

where ϵ is a gradient tolerance. This process is repeated until the optimality criterion is achieved.

The active set is changed only when a free variable hits its bounds during an iteration, or the optimality condition is met for the free variables but not for all variables in IA, the active set. In the latter case, a variable which violates the

optimality condition will be dropped out of IA. For more detail on the Levenberg-Marquardt method, see Levenberg (1944), or Marquardt (1963). For more detailed information on active set strategy, see Gill and Murray (1976).

Since a finite-difference method is used to estimate the Jacobian, for some single precision calculations, an inaccurate estimate of the Jacobian may cause the algorithm to terminate at a noncritical point. In such cases high precision arithmetic is recommended. Also, whenever the exact Jacobian can be easily provided, IMSL routine BCLSJ should be used instead.

Example

The nonlinear least squares problem

$$\min_{x \in \mathbb{R}^2} \frac{1}{2} \sum_{i=1}^{2} f_i(x)^2$$

$$\text{subject to} \quad -2 \le x_1 \le 0.5$$
$$-1 \le x_2 \le 2.$$

where $f_1(x) = 10(x_2 - x_1^2)$, and $f_2(x) = (1 - x_1)$ is solved with an initial guess $(-1.2, 1.0)$, and default values for parameters.

```
C                                 Declaration of variables
      INTEGER    LDFJAC, M, N
      PARAMETER  (LDFJAC=2, M=2, N=2)
C
      INTEGER    IPARAM(7), ITP, NOUT
      REAL       FJAC(LDFJAC,N), FSCALE(M), FVEC(M), ROSBCK,
     &           RPARAM(7), X(N), XGUESS(N), XLB(N), XS(N), XUB(N)
      EXTERNAL   BCLSF, ROSBCK, UMACH
C                                 Compute the least squares for the
C                                 Rosenbrock function.
      DATA XGUESS/-1.2E0, 1.0E0/, XS/2*1.0E0/, FSCALE/2*1.0E0/
      DATA XLB/-2.0E0, -1.0E0/, XUB/0.5E0, 2.0E0/
C                                 All the bounds are provided
      ITP = 0
C                                 Default parameters are used
      IPARAM(1) = 0
C
      CALL BCLSF (ROSBCK, M, N, XGUESS, ITP, XLB, XUB, XS, FSCALE,
     &            IPARAM, RPARAM, X, FVEC, FJAC, LDFJAC)
C                                 Print results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) X, FVEC, IPARAM(3), IPARAM(4)
C
99999 FORMAT (' The solution is ', 2F9.4, //, ' The function ',
     &       'evaluated at the solution is ', /, 18X, 2F9.4, //,
     &       ' The number of iterations is ', 10X, I3, /, ' The ',
```

```
     &       'number of function evaluations is ', I3, /)
      END
C
      SUBROUTINE ROSBCK (M, N, X, F)
      INTEGER    M, N
      REAL       X(N), F(M)
C
      F(1) = 1.0E1*(X(2)-X(1)*X(1))
      F(2) = 1.0E0 - X(1)
      RETURN
      END
```

Output

```
 The solution is      .5000     .2500

 The function evaluated at the solution is
                      .0000     .5000

 The number of iterations is           15
 The number of function evaluations is  22
```

References

Dennis. J. E.. Jr.. and Robert B. Schnabel (1983), *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall. Englewood Cliffs. New Jersey.

Gill. Philip E.. and Walter Murray (1976). *Minimization subject to bounds on the variables*. NPL Report NAC 72. National Physical Laboratory, England.

Levenberg. K. (1944). A method for the solution of certain problems in least squares. *Quarterly of Applied Mathematics*. 2. 164–168.

Marquardt. D. (1963). An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*. 11. 431–441.

IVPRK/DIVPRK (Single/Double precision)

Purpose: Solve an initial-value problem for ordinary differential equations using the Runge-Kutta-Verner fifth-order and sixth-order method.

Usage: CALL IVPRK (IDO, NEQ, FCN, X, XEND, TOL, PARAM, Y)

Arguments

IDO - Flag indicating the state of the computation. (Input/Output)

1	Initial entry
2	Normal reentry
3	Final call to release workspace
4	Return because of interrupt 1
5	Return because of interrupt 2 with step accepted
6	Return because of interrupt 2 with step rejected

Normally, the initial call is made with IDO=1. The routine then sets IDO=2 and this value is then used for all but the last call which is made with IDO=3. This final call is only used to release workspace, which was automatically allocated by the initial call with IDO=1. See Remark 3 for a description of the interrupts.

NEQ - Number of differential equations. (Input)

FCN - User-supplied SUBROUTINE to evaluate functions. The usage is CALL FCN (NEQ, X, Y, YPRIME), where

- NEQ - Number of equations. (Input)
- X - Independent variable. (Input)
- Y - Array of length NEQ containing the dependent variable values. (Input)
- YPRIME - Array of length NEQ containing the values of dY/dX at (X,Y). (Output)

FCN must be declared EXTERNAL in the calling program.

X - Independent variable. (Input/Output)
On input, X supplies the initial value.
On output, X is replaced by XEND unless error conditions arise. See IDO for details.

XEND - Value of X at which the solution is desired. (Input)
XEND may be less than the initial value of X.

TOL - Tolerance for error control. (Input)
An attempt is made to control the norm of the local error such that the global error is proportional to TOL.
More than one run, with different values of TOL, can be

```
        used to estimate the global error.
        Generally, it should not be greater than 0.001.
PARAM - Vector of length 50 containing optional parameters.
        (Input/Output)
        If a parameter is zero then a default value is used.
        The following parameters must be set by the user.
          PARAM            Meaning
            1    HINIT  - Initial value of the step size H.
                          Default: See Algorithm section.
            2    HMIN   - Minimum value of the step size H.
                          Default: 0.0
            3    HMAX   - Maximum value of the step size H.
                          Default: No limit is imposed on the
                          step size.
            4    MXSTEP - Maximum number of steps allowed.
                          Default: 500
            5    MXFCN  - Maximum number of function evaluations
                          allowed.
                          Default: No limit
            6           - Not used.
            7    INTRP1 - If nonzero then return with IDO=4,
                          before each step.
                          See Remark 3.
                          Default: 0.
            8    INTRP2 - If nonzero then return with IDO=5,
                          after every successful step and with
                          IDO=6 after every unsuccessful step.
                          See Remark 3.
                          Default: 0.
            9    SCALE  - A measure of the scale of the problem,
                          such as an approximation to the average
                          value of a norm of the Jacobian along
                          the trajectory.
                          Default: 1.0
           10    INORM  - Switch determining error norm.
                          In the following Ei is the absolute
                          value of an estimate of the error in
                          Y(i), called Yi here.
                          0 - min(absolute error, relative error)
                              = max(Ei/Wi), i=1,2,...,NEQ, where
                              Wi = max(abs(Yi,1.0),
                          1 - absolute error = max(Ei), i=1,2,...
                          2 - max(Ei/Wi), i=1,2,..., where
                              Wi = max(abs(Yi),FLOOR),
                              and FLOOR is PARAM(11).
```

```
                         3 - Euclidian norm scaled by YMAX
                             = sqrt(sum(Ei**2/Wi**2)), where
                             Wi = max(abs(Yi),1.0), for YMAX,
                             see Remark 1.
              11   FLOOR  - Used in the norm computation.
                            Default: 1.0
              12-30       - Not used.
            The following entries in PARAM are set by the program.
              31   HTRIAL - Current trial step size.
              32   HMINC  - Computed minimum step size allowed.
              33   HMAXC  - Computed maximum step size allowed.
              34   NSTEP  - Number of steps taken.
              35   NFCN   - Number of function evaluations used.
              36-50       - Not used.
Y       - Vector of length NEQ of dependent variables.
          (Input/Output)
          On input, Y contains the initial values.  On output,
          Y contains the approximate solution.
```

Remarks

```
1. Automatic workspace usage is
          IVPRK    10*NEQ units, or
          DIVPRK   20*NEQ units.
   Workspace may be explicitly provided, if desired, by use of
   I2PRK/DI2PRK.  The reference is
          CALL I2PRK (IDO, NEQ, FCN, X, XEND, TOL, PARAM, Y,
                      VNORM, WK)
   The additional arguments are as follows:
   VNORM  - User-supplied SUBROUTINE to compute the norm of the
            error.  (Input)
            The routine may be provided by the user, or the IMSL
            routine I3PRK/DI3PRK may be used.
            The usage is
            CALL VNORM (NEQ, V, Y, YMAX, ENORM), where
            NEQ    - Number of equations.  (Input)
            V      - Vector of length NEQ containing the vector whose
                     norm is to be computed.  (Input)
            Y      - Vector of length NEQ containing the values of
                     the dependent variable.  (Input)
            YMAX   - Vector of length NEQ containing the maximum Y
                     values computed so far.  (Input)
            ENORM  - Norm of the vector V.  (Output)
            VNORM must be declared EXTERNAL in the calling program.
   WK     - Work array of length 10*NEQ.  WK must not be changed
```

from the first call with IDO=1 until after the final call with IDO=3.

2. Informational errors

Type	Code	
4	1	Cannot satisfy error condition. TOL may be too small.
4	2	Too many function evaluations needed.
4	3	Too many steps needed. The problem may be stiff.

3. If PARAM(7) is nonzero, the subroutine returns with IDO = 4, and will resume calculation at the point of interruption if reentered with IDO = 4. If PARAM(8) is nonzero, the subroutine will interrupt the calculations immediately after it decides whether or not to accept the result of the most recent trial step. IDO = 5 if the routine plans to accept, or IDO = 6 if it plans to reject. IDO may be changed by the user in order to force acceptance of a step (by changing IDO from 6 to 5) that would otherwise be rejected, or vice versa. Relevant parameters to observe after return from an interrupt are IDO, HTRIAL, NSTEP, NFCN, and Y. Y is the newly computed trial value, accepted or not.

Algorithm

IVPRK finds an approximation to the solution of a system of first-order differential equations of the form $y' = f(x, y)$ with initial conditions. The routine attempts to keep the global error proportional to a user-specified tolerance. The proportionality depends on the differential equation and the range of integration.

IVPRK is efficient for nonstiff systems where the derivative evaluations are not expensive and where the solution is not required at a large number of finely spaced points (as might be required for graphical output).

IVPRK is based on a code designed by T. E. Hull, W. H. Enright and K. R. Jackson (1976, 1977). It uses Runge-Kutta formulas of order five and six developed by J. H. Verner.

Example

Consider a predator-prey problem with rabbits and foxes. Let r be the density of rabbits and let f be the density of foxes. In the absence of any predator-prey interaction the rabbits would increase at a rate proportional to their number, and the foxes would die of starvation at a rate proportional to their number. Mathematically,

$$\begin{aligned} r' &= 2r \\ f' &= -f. \end{aligned}$$

The rate at which the rabbits are eaten by the foxes is $2rf$ and the rate at which the foxes increase, because they are eating the rabbits, is rf. So the model to be solved is

$$
\begin{aligned}
r' &= 2r - 2rf \\
f' &= -f + rf.
\end{aligned}
$$

The initial conditions are $r(0) = 1$ and $f(0) = 3$ over the interval $0 \le t \le 10$.

In the program Y(1) = r and Y(2) = f. Note that the parameter vector is first set to zero (using IMSL routine SSET) and then absolute error control is selected by setting PARAM(10) = 1.0.

The last call to IVPRK with IDO = 3 releases IMSL workspace, that was reserved on the first call to IVPRK. It is not necessary to release the workspace in this example, because the program ends after solving a single problem. The call to release workspace is made as a model of what would be needed if the program included further calls to IMSL routines.

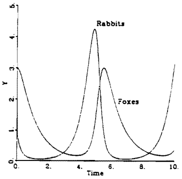
The following plots are the result of using IVPRK with more closely spaced output than what is printed. (The program which does the plotting is not shown.) The second plot is a phase diagram for this system and clearly shows the periodic nature of the solution.

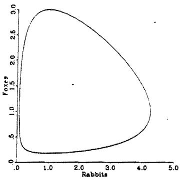
```
      INTEGER    MXPARM, NEQ
      PARAMETER  (MXPARM=50, NEQ=2)
C
      INTEGER    IDO, ISTEP, NOUT
      REAL       FCN, FLOAT, PARAM(MXPARM), T, TEND, TOL, Y(NEQ)
      INTRINSIC  FLOAT
      EXTERNAL   FCN, IVPRK, SSET, UMACH
C
      CALL UMACH (2, NOUT)
C                                 Set initial conditions
      T    = 0.0
      Y(1) = 1.0
      Y(2) = 3.0
C                                 Set error tolerance
      TOL = 0.0005
C                                 Set PARAM to default
      CALL SSET (MXPARM, 0.0, PARAM, 1)
C                                 Select absolute error control
      PARAM(10) = 1.0
C                                 Print header
      WRITE (NOUT,99999)
99999 FORMAT (4X, 'ISTEP', 5X, 'Time', 9X, 'Y1', 11X, 'Y2')
      IDO = 1
      DO 10  ISTEP=1, 10
         TEND = FLOAT(ISTEP)
         CALL IVPRK (IDO, NEQ, FCN, T, TEND, TOL, PARAM, Y)
         WRITE (NOUT,'(I6,3F12.3)') ISTEP, T, Y
   10 CONTINUE
```

```
C                                 Final call to release workspace
      IDO = 3
      CALL IVPRK (IDO, NEQ, FCN, T, TEND, TOL, PARAM, Y)
      END
      SUBROUTINE FCN (NEQ, T, Y, YPRIME)
      INTEGER    NEQ
      REAL       T, Y(NEQ), YPRIME(NEQ)
C
      YPRIME(1) = 2.0*Y(1) - 2.0*Y(1)*Y(2)
      YPRIME(2) = -Y(2) + Y(1)*Y(2)
      RETURN
      END
```

Output

ISTEP	Time	Y1	Y2
1	1.000	078	1 465
2	2.000	085	.578
3	3.000	.292	.250
4	4.000	1.449	187
5	5.000	4.046	1.444
6	6.000	.176	2.256
7	7.000	.066	.908
8	8.000	.148	.367
9	9.000	.655	.188
10	10.000	3.157	.352





References

Hull, T. E., W. H. Enright, and K. R. Jackson (1976). *User's guide for DVERK — A subroutine for solving non-stiff ODEs*. Department of Computer Science Technical Report 100, University of Toronto.

Jackson, K. R., W. H. Enright, and T. E. Hull (1977). *A theoretical criterion for comparing Runge-Kutta formulas*. Department of Computer Science Technical Report 101, University of Toronto.

Basic Linear Algebra Subprograms

The basic linear algebra subprograms, usually called the BLAS, are routines for low-level vector operations such as dot products. Lawson et al. (1979) developed the original set of 38 BLAS routines. The IMSL BLAS collection includes these original 38 routines plus additional routines. The original BLAS are marked with a * in the descriptions.

Programming Notes

The BLAS do not follow the usual IMSL naming conventions. Instead the names consist of a prefix of one or more of the letters 'I', 'S', 'D', 'C' and 'Z', a root name, and sometimes a suffix. For subprograms involving a mixture of data types the output type is indicated by the first prefix letter. The suffix denotes a variant algorithm. The prefix denotes the type of the operation according to the following table:

I	Integer		
S	Real	C	Complex
D	Double	Z	Double complex
SD	Single and double	CZ	Single and double complex
DQ	Double and quadruple	ZQ	Double and quadruple complex

Vector arguments have an increment parameter which specifies the storage space between elements. The correspondence between the vector x and the arguments SX and INCX is

$$x_i = \begin{cases} \texttt{SX((I-1)*INCX+1)} & \text{if } \texttt{INCX} \geq 0 \\ \texttt{SX((I-N)*INCX+1)} & \text{if } \texttt{INCX} < 0. \end{cases}$$

Only positive values of INCX are allowed for operations which have a single vector argument.

The loops in all of the BLAS routines process the vector arguments in order of increasing i. For INCX < 0, this implies processing in reverse storage order.

With the definitions,

$$\begin{aligned} \texttt{MX} &= \max\{1, 1 + (N-1)|\texttt{INCX}|\} \\ \texttt{MY} &= \max\{1, 1 + (N-1)|\texttt{INCY}|\} \\ \texttt{MZ} &= \max\{1, 1 + (N-1)|\texttt{INCZ}|\} \end{aligned}$$

the routine descriptions assume the following FORTRAN declarations:

```
      IMPLICIT INTEGER          (I-N)
      IMPLICIT REAL             S
      IMPLICIT DOUBLE PRECISION D
      IMPLICIT COMPLEX          C
      IMPLICIT DOUBLE COMPLEX   Z
      INTEGER                   IX(MX)
      REAL                      SX(MX), SY(MY), SZ(MZ), SPARAM(5),
     &                          SH(LDH,*)
```

	Integer	Real	Double	Complex	Double Complex	Page
$x_i = a$	ISET	SSET	DSET	CSET	ZSET	1026
$y_i = x_i$	ICOPY	SCOPY	DCOPY	CCOPY	ZCOPY	1026
$x_i = ax_i$		SSCAL	DSCAL	CSCAL	ZSCAL	1026
$a \in \mathbb{R}$				CSSCAL	ZDSCAL	1026
$y_i = ax_i$		SVCAL	DVCAL	CVCAL	ZVCAL	1027
$a \in \mathbb{R}$				CSVCAL	ZDVCAL	1027
$x_i = x_i + a$	IADD	SADD	DADD	CADD	ZADD	1027
$x_i = a - x_i$	ISUB	SSUB	DSUB	CSUB	ZSUB	1027
$y_i = ax_i + y_i$		SAXPY	DAXPY	CAXPY	ZAXPY	1027
$y_i \leftrightarrow x_i$	ISWAP	SSWAP	DSWAP	CSWAP	ZSWAP	1028
$x \cdot y$		SDOT	DDOT	CDOTU	ZDOTU	1028
$\bar{x} \cdot y$				CDOTC	ZDOTC	1028
$x \cdot y$ †		DSDOT		CZDOTU	ZQDOTU	1028
$\bar{x} \cdot y$				CZDOTC	ZQDOTC	1028
$a + x \cdot y$ †		SDSDOT	DQDDOT	CZUDOT	ZQUDOT	1028
$a + \bar{x} \cdot y$				CZCDOT	ZQCDOT	1028
$b + x \cdot y$ †		SDDOTI	DQDOTI	CZDOTI	ZQDOTI	1029
ACC$+b + x \cdot y$ †		SDDOTA	DQDOTA	CZDOTA	ZQDOTA	1029
$z_i = x_i y_i$		SHPROD	DHPROD			1029
$\sum x_i y_i z_i$		SXYZ	DXYZ			1029
$\sum x_i$	ISUM	SSUM	DSUM			1029
$\sum \lvert x_i \rvert$		SASUM	DASUM	SCASUM	DZASUM	1030
$\lVert x \rVert_2$		SNRM2	DNRM2	SCNRM2	DZNRM2	1030
$\prod x_i$		SPRDCT	DPRDCT			1030
$i: \ x_i = \min_j x_j$	IIMIN	ISMIN	IDMIN			1030
$i: \ x_i = \max_j x_j$	IIMAX	ISMAX	IDMAX			1030
$i: \ \lvert x_i \rvert = \min_j \lvert x_j \rvert$		ISAMIN	IDAMIN	ICAMIN	IZAMIN	1031
$i: \ \lvert x_i \rvert = \max_j \lvert x_j \rvert$		ISAMAX	IDAMAX	ICAMAX	IZAMAX	1031
Construct Given's rotation		SROTG	DROTG			1031
Apply Given's rotation		SROT	DROT	CSROT	ZDROT	1032
Construct modified Given's transform		SROTMG	DROTMG			1032
Apply modified Given's transform		SROTM	DROTM	CSROTM	ZDROTM	1033
Construct Householder transform		SHOUTR	DHOUTR			1034
Apply Householder transform		SHOUAP	DHOUAP			1034

†Higher precision accumulation used.

```
      DOUBLE PRECISION        DX(MX), DY(MY), DZ(MZ), DPARAM(5),
     &                        DH(LDH,*)
      DOUBLE PRECISION        DACC(2), DZACC(4)
      COMPLEX                 CX(MX), CY(MY)
      DOUBLE COMPLEX          ZX(MX), ZY(MY)
```

Since FORTRAN 77 does not include the type DOUBLE COMPLEX, routines with DOUBLE COMPLEX arguments are not available for all systems. Some systems use the declaration COMPLEX*16 instead of DOUBLE COMPLEX.

The set of BLAS routines are summarized by the table on page 1025. Routines marked with a dagger (†) in the table use higher precision accumulation.

Set a Vector to a Constant Value

```
      CALL ISET (N, IA, IX, INCX)
      CALL SSET (N, SA, SX, INCX)
      CALL DSET (N, DA, DX, INCX)
      CALL CSET (N, CA, CX, INCX)
      CALL ZSET (N, ZA, ZX, INCX)
```

These subroutines set $x_i = a$ for $i = 1, 2, \ldots, N$. If $N \leq 0$ then the routines return immediately.

Copy a Vector

```
      CALL ICOPY (N, IX, INCX, IY, INCY)
*     CALL SCOPY (N, SX, INCX, SY, INCY)
*     CALL DCOPY (N, DX, INCX, DY, INCY)
*     CALL CCOPY (N, CX, INCX, CY, INCY)
      CALL ZCOPY (N, ZX, INCX, ZY, INCY)
```

These subroutines set $y_i = x_i$ for $i = 1, 2, \ldots, N$. If $N \leq 0$ then the routines return immediately.

Scale a Vector

```
*     CALL SSCAL (N, SA, SX, INCX)
*     CALL DSCAL (N, DA, DX, INCX)
*     CALL CSCAL (N, CA, CX, INCX)
      CALL ZSCAL (N, ZA, ZX, INCX)
*     CALL CSSCAL (N, SA, CX, INCX)
      CALL ZDSCAL (N, DA, ZX, INCX)
```

These subroutines set $x_i = ax_i$ for $i = 1, 2, \ldots, N$. If $N \leq 0$ then the routines return immediately.

LIST OF REFERENCES

1. Reno, M. M., "Modeling Transient Thermal Behavior in a Thrust Vector Control Jet Vane", Masters Thesis, Department of Mechanical Engineering, Naval Postgraduate School, Monterey, California, December 1988.

2. Danielson, A. O., "Inverse Heat Transfer Studies and the Effects of Propellant Aluminum on TVC Jet Vane Heating and Erosion", Naval Weapons Center, China Lake, California, July 1990.

3. Danielson, A. O. and Driels, M. R., "Testing and Analysis of Heat Transfer in Materials Exposed to Non-metallized HTPB Propellant", Department of Mechanical Engineering, Naval Postgraduate School, Monterey, California, November 1992.

4. Parker, G. K., "Heat Transfer Parametric System Identification", Masters Thesis, Department of Mechanical Engineering, Naval Postgraduate School, Monterey, California, June 1993.

5. Nunn, R. H., "Jet Vane Modeling Development and Evaluation", Final Report from VRC Corporation, Monterey, California, January 1990.

6. Driels, M. R., "Heat Transfer Parametric Identification", Final Report FY92, Department of Mechanical Engineering, Naval Postgraduate School, Monterey, California, 1992.

7. Nunn, R. H. and Kelleher, M. D., "Jet Vane Heat Transfer Modeling", Research Report, Department of Mechanical Engineering, Naval Postgraduate School, Monterey, California, October 1986.

8. Hatzenbuehler, M. A., "Modeling of Jet Vane heat Transfer Characteristics and Simulation of Thermal Response", Masters Thesis, Department of Mechanical Engineering, Naval Postgraduate School, Monterey, California, June 1988.

9. Rohsevow, W. M. and Choi, H. Y., "Heat, Mass and Momentum Transfer", Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1961.

10. Danielson, A. O. and Figueiredo, W., "Erosion and Heating Correlations for Tungstan Subscale and Full-scale Thrust Vector Control (TVC) Vanes Exposed to Aluminized Propellant", Naval Air Warfare Center Weapons Division, China Lake, California, November 1992.

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Department Chairman, Code ME Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93943-5002	1
4.	Professor Morris R. Driels Code ME/DR Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93943-5002	2
5.	LT Steven R. Gardner P.O. Box 188 Gilmanton, New Hampshire 03237	2
6.	Naval Engineering Curricular Officer, Code 34 Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93943-5002	1
7.	A. Danielson Thermal Structures Branch Naval Air Weapons Center Code C2891 China Lake, California 93555	2